

UNIVERSITY WÜRZBURG

DOCTORAL THESIS

---

# Linkable Technical Documentation

---

*Author:*  
Sebastian FURTH

*Supervisors:*  
PD Dr. Joachim BAUMEISTER  
Prof. Dr. Frank PUPPE

*A thesis submitted in fulfillment of the requirements  
for the degree of Dr. rer. nat.*

*in the*

Faculty of Mathematics and Computer Science  
Institute of Computer Science

2018-08-10





UNIVERSITY WÜRZBURG

*Abstract*Faculty of Mathematics and Computer Science  
Institute of Computer Science

Dr. rer. nat.

**Linkable Technical Documentation**

by Sebastian FURTH

The success of semantic systems has been proven over the last years. Nowadays, Linked Data is the driver for the rapid development of ever new intelligent systems. Especially in enterprise environments semantic systems successfully support more and more business processes. This is especially true for after sales service in the mechanical engineering domain. Here, service technicians need effective access to relevant technical documentation in order to diagnose and solve problems and defects. Therefore, the usage of semantic information retrieval systems has become the new system metaphor. Unlike classical retrieval software Linked Enterprise Data graphs are exploited to grant targeted and problem-oriented access to relevant documents. However, huge parts of legacy technical documents have not yet been integrated into Linked Enterprise Data graphs. Additionally, a plethora of information models for the semantic representation of technical information exists. The semantic maturity of these information models can hardly be measured.

This thesis motivates that there is an inherent need for a self-contained semantification approach for technical documents. This work introduces a maturity model that allows to quickly assess existing documentation. Additionally, the approach comprises an abstracting semantic representation for technical documents that is aligned to all major standard information models. The semantic representation combines structural and rhetorical aspects to provide access to so called Core Documentation Entities. A novel and holistic semantification process describes how technical documents in different legacy formats can be transformed to a semantic and linked representation.

The practical significance of the semantification approach depends on tools supporting its application. This work presents an accompanying tool chain of semantification applications, especially the semantification framework CAPLAN that is a highly integrated development and runtime environment for semantification processes. The complete semantification approach is evaluated in four real-life projects: in a spare part augmentation project, semantification projects for earth moving technology and harvesting technology, as well as an ontology population project for special purpose vehicles. Three additional case studies underline the broad applicability of the presented ideas.



## *Acknowledgements*

First and fore-most I want to thank my main supervisor Joachim Baumeister for providing me the opportunity to pursue and evolve the topic of semantifying technical documents. From the very beginning you always provided me a lot of real-world problems and offered an open door to discuss them under practical and scientific aspects. You have always been a great and open-minded supporter and motivator. I really enjoyed our paper-collaboration including the valuable discussions that helped this work progress. Many thanks also to my co-supervisor Frank Puppe. In all stadiums of this work you provided a lot of helpful feedback and astute questions that helped this project succeed. You also supported me by answering all the smaller and bigger administrative questions arising from me being an external doctoral candidate.

A special thanks goes to my employer *denkbares* for supporting my research and the writing of this thesis. Thanks also to all my kind colleagues at *denkbares*, especially Markus Friedrich, Sandrina Nothnick, Stefan Plehn, Jochen Reutelshöfer and Albrecht Striffler. Working with you has always been a pleasure and you supported me a lot—especially when times were not so good for me. Thanks especially to my colleague Volker Belli. You were always open to intense discussions and really pushed this work with ever new impulses and ideas. I also want to give a special thank you to Alex Legler not only for reading and polishing parts of this thesis but also for the great time in our office room that we have been sharing for more than four years. Thanks also to Gerd Siebert who pointed me to the VW ILTIS data set.

I thank my family for their support, advices and open ears, especially my mother Edeltraud, my stepfather Helmut, my father Florian and my sister Bianca. Last but definitely not least, I thank Christine for being an anchor in my life: no one understands me better and you always found the right words that kept up my belief and motivation to finish this work. Thank you!

Without the support of all these persons, this work would not have been possible.



Sebastian Furth



# Contents

<b>Abstract</b>	<b>iii</b>
<b>Acknowledgements</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Research Goal and Context . . . . .	2
1.2 Approach . . . . .	3
1.2.1 5-STAR Technical Documentation . . . . .	3
1.2.2 5-STAR Semantification . . . . .	4
1.2.3 Domain Knowledge as the Regulating Screw . . . . .	6
1.2.4 Unified Pipelines and Integrated Environments . . . . .	6
1.3 Results . . . . .	6
1.4 Structure of this Work . . . . .	8
<b>2 Accessing Technical Documents</b>	<b>11</b>
2.1 Technical Documentation . . . . .	12
2.1.1 Characteristics of Technical Documentation . . . . .	12
2.1.2 Types of Technical Documentation . . . . .	12
2.2 Semantic Technologies . . . . .	14
2.2.1 In a Nutshell . . . . .	14
2.2.2 Types of Ontologies . . . . .	15
2.2.3 Linked Data . . . . .	16
2.2.4 Linked Enterprise Data . . . . .	16
2.3 Text Analytics for Linked Data Integration . . . . .	19
2.3.1 Linked Data Lifecycle . . . . .	19
2.3.2 Linked Data Integration in a Nutshell . . . . .	20
2.3.3 Text Analytics as Part of Linked Data Integration . . . . .	21
2.4 Semantic Information Retrieval . . . . .	27
<b>3 Deep Semantics for Technical Documents</b>	<b>31</b>
3.1 Overview . . . . .	32
3.2 5-STAR Technical Documentation . . . . .	32
3.2.1 1-STAR: Electronic Format . . . . .	34
3.2.2 2-STAR: Structured Content . . . . .	35
3.2.3 3-STAR: Atomic Modules . . . . .	36
3.2.4 4-STAR: Information Typed Modules . . . . .	37
3.2.5 5-STAR: Linked Information Units . . . . .	37
3.3 The Technical Knowledge Ontology . . . . .	39
3.3.1 Structural Representation . . . . .	40
3.3.2 Information Types . . . . .	43
3.3.3 Core Documentation Entities . . . . .	45
3.4 Information Models for Technical Documents . . . . .	51
3.4.1 PI-Mod . . . . .	51

3.4.2	iiRDS . . . . .	56
3.4.3	DITA . . . . .	61
3.4.4	DocBook . . . . .	65
3.4.5	S1000D . . . . .	70
3.5	Summary and Contributions . . . . .	77
<b>4</b>	<b>Semantification of Technical Documents</b>	<b>79</b>
4.1	Overview . . . . .	80
4.2	Preparatory Steps . . . . .	81
4.2.1	Data Selection . . . . .	81
4.2.2	Data Cleaning . . . . .	82
4.3	1-STAR . . . . .	83
4.3.1	Information Systems Use Case . . . . .	83
4.3.2	Problem Description . . . . .	83
4.3.3	Document Layout Analysis . . . . .	84
4.3.4	Electronic Formats . . . . .	88
4.3.5	Practical Recommendations . . . . .	94
4.3.6	Related Work . . . . .	95
4.4	2-STAR . . . . .	96
4.4.1	Information Systems Use Case . . . . .	96
4.4.2	Problem Description . . . . .	96
4.4.3	Logical Document Structure Recovery and Analysis . . . . .	98
4.4.4	Interactive Knowledge Acquisition . . . . .	104
4.4.5	Knowledge Acquisition in Practice . . . . .	105
4.4.6	Recommended Formats . . . . .	106
4.4.7	Practical Recommendations . . . . .	106
4.4.8	Related Work . . . . .	107
4.5	3-STAR . . . . .	108
4.5.1	Information Systems Use Case . . . . .	108
4.5.2	Problem Description . . . . .	108
4.5.3	Macro Structure Recovery . . . . .	109
4.5.4	Deduplication . . . . .	110
4.5.5	Alignment . . . . .	114
4.5.6	Recommended Formats . . . . .	116
4.5.7	Practical Recommendations . . . . .	117
4.5.8	Related Work . . . . .	118
4.6	4-STAR . . . . .	119
4.6.1	Information Systems Use Case . . . . .	119
4.6.2	Problem Description . . . . .	119
4.6.3	Automatic Document Classification . . . . .	120
4.6.4	Ground Truth and Practical Alternatives . . . . .	122
4.6.5	Recommended Formats . . . . .	123
4.6.6	Practical Recommendations and Related Work . . . . .	124
4.7	5-STAR . . . . .	125
4.7.1	Information Systems Use Case . . . . .	125
4.7.2	Problem Description . . . . .	125
4.7.3	Explicit Semantic Analysis . . . . .	128
4.7.4	Probabilistic Subject Analysis . . . . .	131
4.7.5	Interactive Review . . . . .	134
4.7.6	Sources of Ontologies . . . . .	136
4.7.7	Recommended Formats . . . . .	137



4.7.8	Practical Recommendations . . . . .	138
4.7.9	Related Work . . . . .	138
4.8	Summary and Contributions . . . . .	139
<b>5</b>	<b>Implementation of a Semantification Architecture</b>	<b>141</b>
5.1	Overview . . . . .	142
5.2	Distinction: Existing 1-STAR Semantification Tools . . . . .	142
5.3	Using Semantification Methods: CAPLAN . . . . .	143
5.3.1	Requirements . . . . .	143
5.3.2	Architecture . . . . .	144
5.3.3	Implementation . . . . .	146
5.4	Developing Semantification Knowledge:	
TEKNO Studio . . . . .		152
5.4.1	Workflow . . . . .	153
5.4.2	Implementation . . . . .	154
5.5	Reviewing Semantification Results: Review Tool . . . . .	156
5.6	Summary and Contributions . . . . .	158
<b>6</b>	<b>Hands On: Semantification by Example</b>	<b>161</b>
6.1	Source Data and Semantification Goal . . . . .	162
6.2	1-STAR: Document Layout Analysis . . . . .	163
6.3	2-STAR: Logical Document Structure Recovery . . . . .	164
6.3.1	TEKNO Studio: Knowledge Acquisition . . . . .	164
6.3.2	Classification Knowledge . . . . .	166
6.3.3	Output . . . . .	168
6.4	3-STAR: Macro Structure Recovery . . . . .	170
6.5	4-STAR: Information Typing . . . . .	172
6.6	5-STAR: Semantic Annotation . . . . .	174
6.6.1	Bike Ontology . . . . .	175
6.6.2	Entity Recognition . . . . .	175
6.6.3	Probabilistic Subject Analysis . . . . .	176
6.6.4	Output . . . . .	178
6.7	A Demonstration of Improved Accessibility . . . . .	180
<b>7</b>	<b>Experiences</b>	<b>181</b>
7.1	Overview . . . . .	182
7.2	S1000D Bike . . . . .	182
7.2.1	Introduction . . . . .	182
7.2.2	Goals and Application Scenarios . . . . .	182
7.2.3	Data Set Description . . . . .	183
7.2.4	The Semantification Process . . . . .	183
7.2.5	Knowledge Resources . . . . .	185
7.2.6	Evaluation . . . . .	188
7.2.7	System Use . . . . .	192
7.2.8	Summary . . . . .	194
7.3	PI-Fan . . . . .	195
7.3.1	Introduction . . . . .	195
7.3.2	Goals and Application Scenarios . . . . .	195
7.3.3	Data Set Description . . . . .	195
7.3.4	The Semantification Process . . . . .	195
7.3.5	Knowledge Resources . . . . .	196

7.3.6	Evaluation . . . . .	196
7.3.7	System Use . . . . .	197
7.3.8	Summary . . . . .	198
7.4	Augmenting Spare Part Catalogues . . . . .	199
7.4.1	Introduction . . . . .	199
7.4.2	Goals and Application Scenario . . . . .	199
7.4.3	Data Set Description . . . . .	199
7.4.4	The Semantification Process . . . . .	199
7.4.5	System Use . . . . .	201
7.4.6	Summary . . . . .	201
7.5	Earth Moving Technology . . . . .	203
7.5.1	Introduction . . . . .	203
7.5.2	Goals and Application Scenario . . . . .	203
7.5.3	Data Set Description . . . . .	203
7.5.4	The Semantification Process . . . . .	204
7.5.5	Knowledge Resources . . . . .	205
7.5.6	System Use . . . . .	206
7.5.7	Summary . . . . .	207
7.6	Special Purpose Vehicles: Ontology Population . . . . .	209
7.6.1	Introduction . . . . .	209
7.6.2	Goals and Application Scenario . . . . .	209
7.6.3	Data Set Description . . . . .	209
7.6.4	Ontology Population Architecture . . . . .	209
7.6.5	System Use . . . . .	213
7.6.6	Summary . . . . .	214
7.7	Harvesting Technology . . . . .	215
7.7.1	Introduction . . . . .	215
7.7.2	Goals and Application Scenario . . . . .	215
7.7.3	Data Set Description . . . . .	215
7.7.4	The Semantification Process . . . . .	216
7.7.5	Knowledge Resources . . . . .	217
7.7.6	Evaluation . . . . .	217
7.7.7	System Use . . . . .	219
7.7.8	Summary . . . . .	220
7.8	VW ILTIS . . . . .	221
7.8.1	Introduction . . . . .	221
7.8.2	Goals and Application Scenario . . . . .	221
7.8.3	Data Set Description . . . . .	221
7.8.4	The Semantification Process . . . . .	222
7.8.5	Knowledge Resources . . . . .	223
7.8.6	System Use . . . . .	224
7.8.7	Summary . . . . .	226
<b>8</b>	<b>Conclusion</b> . . . . .	<b>227</b>
8.1	Summary . . . . .	228
8.1.1	Accessing Technical Documents . . . . .	228
8.1.2	Deep Semantics for Technical Documents . . . . .	228
8.1.3	Semantification of Technical Documents . . . . .	229
8.1.4	Implementation of a Semantification Architecture . . . . .	229
8.1.5	Practical Experiences . . . . .	230
8.2	Discussion . . . . .	232

8.2.1	Self-Contained Semantification Approach . . . . .	232
8.2.2	On the Exploitation of Table of Contents . . . . .	234
8.2.3	Evaluation of the Practicability in Four Real-World Projects . . . . .	235
8.3	Outlook . . . . .	236
8.3.1	Fully Integrated Tool Chain . . . . .	236
8.3.2	Extending Core Documentation Entity Catalogue . . . . .	236
8.3.3	Architecture Definition . . . . .	236
8.3.4	Automatic Thesaurus Learning . . . . .	237
8.3.5	Test, Analyze, Check . . . . .	237
8.3.6	Application to other Domains . . . . .	237

## **Bibliography**

**239**



# List of Figures

2.1	“Visualization of the common, layered Semantic Web technology stack” by jsaiya / CC BY 3.0. . . . .	14
2.2	“Linking Open Data cloud diagram 2017”, by Andrejs Abele, John P. McCrae, Paul Buitelaar, Anja Jentzsch and Richard Cyganiak. ( <a href="http://lod-cloud.net">http://lod-cloud.net</a> ). . . . .	17
2.3	“Linked Data Lifecycle”, by Auer et al. [4]. . . . .	20
2.4	Excerpt of a stock list showing the contained terms. . . . .	24
2.5	Term city showing a term source with high coverage. . . . .	25
2.6	Term city showing a term source with low coverage. . . . .	26
2.7	Architecture of a Semantic Search Engine. . . . .	29
3.1	The levels of the 5-STARs maturity schema for technical documentation. . . . .	33
3.2	Overview of the TEKNO ontology. . . . .	39
3.3	Overview of the different types of structural components. . . . .	40
3.4	Examples of micro structures. . . . .	41
3.5	Examples of macro structures. . . . .	42
3.6	Examples of information typed structures. . . . .	45
3.7	Overview of the PI-Mod information model. . . . .	52
3.8	Mapping of the PI-Mod information model to the TEKNO ontology. . . . .	55
3.9	Overview of the iiRDS information model. . . . .	57
3.10	Directory nodes for the representation of document structures. . . . .	58
3.11	Mapping of the iiRDS information model to the TEKNO ontology. . . . .	60
3.12	Overview of the DITA information model. . . . .	62
3.13	Mapping of the DITA information model to the TEKNO ontology. . . . .	64
3.14	Overview of the DITA information model. . . . .	66
3.15	Mapping of the DocBook information model to the TEKNO ontology. . . . .	69
3.16	Overview of the S1000D information model. . . . .	71
3.17	S1000D data module code. . . . .	74
3.18	Mapping of the S1000D information model to the TEKNO ontology. . . . .	75
4.1	Prioritization of documents on priority lists. . . . .	81
4.2	Skew estimation for a single page. . . . .	84
4.3	Separation of text and non-text zones. . . . .	86
4.4	Tokenization for Nano Structure Recovery. . . . .	86
4.5	Segmentation for Micro Structure Recovery. . . . .	87
4.6	Micro structures. . . . .	97
4.7	Overview of the recovery and classification phases. . . . .	99
4.8	Document Structure Ontology according to the DocBook information model. . . . .	99
4.9	Set-Covering model for the classification of the classes $C_1$ and $C_2$ , using the parameters $A_1, A_2, A_3$ and $A_4$ . . . . .	100
4.10	Set-covering model for the classification of the classes <i>Heading 1</i> and <i>Heading 2</i> , using textual parameters. . . . .	101

4.11	TEKNO Studio: Tool support for interactive knowledge acquisition. . . . .	105
4.12	Macro Structure Recovery from Micro Structures. . . . .	110
4.13	Micro Structure Sequence Similarity determination. . . . .	114
4.14	Macro Structure Hierarchy Alignment. . . . .	115
4.15	Information Type Classification. . . . .	120
4.16	The elements of technical documentation. . . . .	126
4.17	Transformation of an information unit to a bag of terms using entity recognition methods. . . . .	127
4.18	Determination of the main subjects on basis of the discovered termi- nology. . . . .	129
4.19	An excerpt of a semantic interpreter showing the gear stick example. . .	129
4.20	A tool for the manual review of semantic annotations, containing the hierarchical segmentation (left), a visual report (top right) and a detail view (bottom right). . . . .	135
5.1	Key components of the Semantification Architecture. . . . .	145
5.2	CAPLAN: Overview of supported data node types. . . . .	148
5.3	CAPLAN: Modules handling Term Data Nodes. . . . .	149
5.4	CAPLAN: Modules handling Document Data Nodes. . . . .	150
5.5	CAPLAN: Error in Semantification Module. . . . .	150
5.6	CAPLAN: Library of Semantification Modules. . . . .	151
5.7	CAPLAN: Configuration of a Semantification Module. . . . .	151
5.8	CAPLAN: Parametrization of a Semantification Module. . . . .	152
5.9	TEKNO: Blockification and Block Selection. . . . .	154
5.10	TEKNO: Create a Set-Covering Model. . . . .	155
5.11	TEKNO: Preview of classification results. . . . .	155
5.12	Review Tool: Overview. . . . .	156
5.13	Review Tool: Macro Structure Hierarchy. . . . .	157
5.14	Review Tool: Visual Report. . . . .	157
5.15	Review Tool: Editing Macro Structure Titles. . . . .	158
5.16	Review Tool: Annotation Editing supported by Semantic Typing. . . .	158
6.1	Hands On: Source PDF. . . . .	162
6.2	Hands On: 1-STAR results. . . . .	164
6.3	Hands On: Typed Micro Structures (1/2). . . . .	165
6.4	Hands On: Typed Micro Structures (2/2). . . . .	166
6.5	Hands On: 2-STAR results. . . . .	169
6.6	Hands On: 3-STAR results from 2-STAR input. . . . .	170
6.7	Hands On: 4-STAR information typing. . . . .	173
6.8	Hands On: 5-STAR semantic annotation. . . . .	178
6.9	Hands On: Improved Accessibility in Service Mate. . . . .	180
7.1	S1000D: Data Module Semantification Architecture. . . . .	184
7.2	S1000D: PDF Semantification Architecture. . . . .	184
7.3	S1000D: Serialized Semantic Interpreter. . . . .	185
7.4	S1000D: Structure of the extracted Ontology. . . . .	186
7.5	S1000D: Extracted Ontology. . . . .	186
7.6	S1000D: Set-Covering Model in TEKNO Studio. . . . .	187
7.7	S1000D Evaluation Architecture. . . . .	188
7.8	S1000D: Semantification Pipeline in CAPLAN. . . . .	193

7.9	S1000D: Semantification Result in the information system “Service Mate” . . . . .	193
7.10	PI-Fan: PDF Semantification Architecture. . . . .	196
7.11	PI-Fan: 2-STAR knowledge acquisition in TEKNO Studio. . . . .	197
7.12	Parts Catalogue: PDF Semantification Architecture. . . . .	200
7.13	Parts Catalogue: Semantification KPIs. . . . .	200
7.14	Parts Catalogue: Semantification Result. . . . .	202
7.15	Earth Moving Technology: Semantification Architecture. . . . .	204
7.16	Earth Moving Technology: Extraction. . . . .	205
7.17	Earth Moving Technology: Semantification Report. . . . .	205
7.18	Earth Moving Technology: 2-STAR knowledge acquisition in TEKNO Studio (1/2). . . . .	207
7.19	Earth Moving Technology: 2-STAR knowledge acquisition in TEKNO Studio (2/2). . . . .	207
7.20	Special Purpose Vehicles: Ontology Population Architecture. . . . .	210
7.21	Special Purpose Vehicles: Document Components and Concepts. . . . .	210
7.22	Partial hierarchies. . . . .	211
7.23	Unified concepts. . . . .	211
7.24	Concept hierarchy. . . . .	212
7.25	Special Purpose Vehicles: Exported ontology information in KnowWE. . . . .	213
7.26	Special Purpose Vehicles: Resulting data set in Service Mate. . . . .	214
7.27	Harvesting Technology: Semantification Architecture. . . . .	216
7.28	Harvesting Technology: Evaluation Architecture. . . . .	218
7.29	VW ILTIS: Scanned Book Pages. . . . .	222
7.30	VW ILTIS: Semantification Architecture. . . . .	222
7.31	VW ILTIS: Spare Parts Catalogue providing Ontological Knowledge. . . . .	223
7.32	VW ILTIS: Extracted Ontology. . . . .	224
7.33	VW ILTIS: Semantification Pipeline in CAPLAN. . . . .	225
7.34	VW ILTIS: Semantification Result (1/2). . . . .	225
7.35	VW ILTIS: Semantification Result (2/2). . . . .	226





# List of Tables

3.1	1-STAR: Maturity Fact Sheet . . . . .	35
3.2	2-STAR: Maturity Fact Sheet . . . . .	36
3.3	3-STAR: Fact Sheet . . . . .	36
3.4	4-STAR: Fact Sheet . . . . .	37
3.5	5-STAR: Fact Sheet . . . . .	38
3.6	PI-Mod 5-STAR rating. . . . .	56
3.7	iiRDS 5-STAR rating. . . . .	61
3.8	DITA 5-STAR rating. . . . .	65
3.9	DocBook 5-STAR rating. . . . .	70
3.10	S1000D information codes and descriptions. . . . .	73
3.11	S1000D 5-STAR rating. . . . .	76
3.12	Summary of 5-STAR ratings for the information models PI-Mod, iiRDS, DITA, DocBook, and S1000D. . . . .	77
6.1	Hands On: Information Type Mapping. . . . .	172
6.2	Hands On: Recognized Entities with Confidences. . . . .	176
6.3	Hands On: Recognized Entities with Confidences. . . . .	177
6.4	Hands On: Topic Probabilities. . . . .	178
7.1	S1000D: Macro Structure Recovery Costs . . . . .	190
7.2	S1000D: Macro Structure Recovery Costs . . . . .	191
7.3	S1000D: 5-STAR Accuracy. . . . .	192
7.4	Harvesting Technology: Precision, Recall, F-Measure and Corrections. . . . .	219
7.5	Harvesting Technology: Measuring the correction effort. . . . .	219



*To all people suffering from depressions.*

*Be patient and never give up! Accept the disease as part of your  
life's journey. In the end, everything what you've been through  
will have made you stronger.*



## Chapter 1

# Introduction

*A linked system [is] the next logical step. [...] In this way, documents on similar topics are indirectly linked, through their key concepts. A keyword search then becomes a search starting from a small number of named nodes, and finding nodes which are close to all of them.*

---

*Sir Tim Berners-Lee, 1989*

## 1.1 Research Goal and Context

Now, 30 years later, we perceive a vast amount of technologies that have been evolved from Berners-Lee's vision. Under the patronage of the World Wide Web Consortium (W3C) numerous specifications and standards have been developed and published. These works are the basis for a large spectrum of semantic technologies that are ranging from knowledge representations over query languages to automatic reasoning. Together they enable Linked Data solutions that find more and more application areas.

One of these application areas is the field of *technical service*. Recently, the mechanical engineering domain has been confronted with a dramatical increase of machine complexity. In the past, machines typically could be simply maintained and repaired using mechanical tools. Today's machines, however, are designed as a combination of mechanical components, electrics, hydraulics, and electronics. Hence, service technicians need an increased competence for their repair and maintenance tasks. As a consequence, the technical documentation became a fundamental information source for service technicians in their daily work. The documentation for a single machine, however, easily comprises up to 10,000 pages. Thus, service technicians need fast and focused access methods to handle the massive volumes of technical documents. In order to provide efficient customer support, the accessibility of information has become a critical success factor.[80]

Semantic Search emerged as the new system paradigm for the access of technical documentation. In contrast to traditional search technologies it operates on semantic information in Linked (Enterprise) Data graphs. However, the incorporation of these technologies in enterprise applications has just started and only a small amount of technical documentation has already been integrated into respective graphs. Thus, the service staff usually still has to deal with large quantities of rather loosely organized legacy information. Examples of such information include operation manuals, installation guides, and repair manuals. In order to make legacy information accessible for semantic technologies, links between the textual content and the Linked Enterprise Data graph must be created. Creating these instances in a manual step requires an in-depth analysis of the original content by humans, which is usually error-prone, time-consuming, and cost-intensive.

In recent years, however, *Text Analytics* approaches have been adapted in order to automate the integration of textual content into Linked Data graphs. Well known examples include Document Structure Recovery, Topic Modeling, and particular Information Extraction tasks for Ontology Learning and Population from text. In every of these fields established and well-performing approaches exist. However, this thesis shows that implementing tailored solutions for technical documents is beneficial. This is motivated by the fact that most state-of-the-art approaches rely on supervised

Machine Learning techniques, which usually require a sufficient amount of training data for decent results. In real-world scenarios such training data is usually not available and the creation under the cost-benefit ratio not economic. Additionally tailored solutions can outperform state-of-the-art approaches by exploiting the characteristics of technical documents. This outperformance can decide about the actual success and application in the field. Respective characteristics include the structure of the content or redundancy aspects through multilingualism and large data volumes. However, such alternative solutions also have to cope with new challenges arising from technical documents characteristics like highly specialized terminologies that make standard dictionaries inapplicable.

This work specifies and approaches the problem of integrating technical documents in Linked Enterprise Data graphs in detail. Therefore, two main research gaps get addressed:

**1. Missing Semantic Representation:**

The first research gap approaches the fact that there is no commonly agreed semantic representation for technical documents. Although a lot of prior work exists in the field of semantic publishing it does not fit the domain of technical documents. Existing information models for technical documents, on the other hand, do not max out the capabilities of semantic representations.

**2. Missing Semantification Process:**

The second research gap is concerned with the fact that a holistic approach for the transformation of legacy technical documents to a corresponding semantic representation does not exist. Existing methods tackle subtasks, but do not consider special features of technical documents. Additionally, most methods lack a comprehensive evaluation methodology.

The work results in Linkable Technical Documentation that is fully integrated in Linked Enterprise Data graphs, compatible with existing W3C standards and allows for effective access to relevant information.

## 1.2 Approach

The presented approach tackles a series of existing research gaps. As a fundamental basis this work presents a novel semantic representation for technical documents. This representation is founded on thorough analysis of large technical corpora, insights from semantic representations for other problem domains and existing standards for writing technical documents. Unfortunately, only small amounts of technical documents are already available in a format that can be easily transformed to a semantic representation. Therefore, a holistic semantification approach shows how legacy documents can be transformed to a semantic representation. Both, the semantic representation and the semantification process has been successfully applied in a series of industrial projects. The industrial application has been accompanied by the development of tools that support the semantification task. The following sections give a brief overview of the different aspects of the approach that is described in this thesis.

### 1.2.1 5-STAR Technical Documentation

Although the insufficient accessibility is an obvious shortcoming of today's technical documentation the problem has not been formally specified yet. Thus, the thesis starts with the introduction of a novel assessment schema for technical documents.

The schema lists a number of objective quality criteria building on each other. For each matching criterion one star is given; that way the maturity of documentation data can range from one star (electronic format) to five stars (Linkable Technical Documentation). This schema is inspired by the idea of evaluating the quality of data in the linked open data cloud [16, 100], and was adapted to the special needs of technical documentation. The schema also serves as a central theme for the remainder of the thesis, as the thesis is structured as a journey that covers the transformation of legacy data (1 star) to Linkable Technical Documents (5 stars). The destination of the 5-star journey is a deep semantic representation for technical documents — Linkable Technical Documents. Thus, the first part of the thesis introduces a novel and deep semantic representation that covers the characteristics of technical documents and allows for effective access to relevant information. The representation builds upon established aspects from the semantic publishing community, i.e. structural (e.g. paragraphs, sections, sentences) and rhetorical elements (e.g. discourse elements / sections like Motivation, Problem Statement or Discussion). However, research has shown that existing work concentrates on scientific articles that can hardly be adapted to the special features of technical documents. Hence, in the scope of this thesis large corpora of technical documents have been analyzed in order to derive characteristic structural (e.g. procedures) and rhetorical (e.g. repair) elements. Following the 5-star idea, the thesis additionally and particularly considered established (de-facto) standards for technical documents (e.g. OASIS DocBook, OASIS DITA, ASD S1000D) during this analysis phase.

The analysis has shown that existing vocabularies often concentrate on either the structural or the rhetorical elements of documents. However, the combination of both the rhetorical and the structural elements allows for the logical deduction of relevant aspects. For instance combining “repair” (rhetorical) and “procedure” (structural) elements allows for the derivation and hence the direct access to repair procedures. The thesis gives a comprehensive description of such *Core Documentation Entities*. Additionally, the thesis introduces a collection of corresponding logical expressions that facilitate the derivation of these Core Documentation Entities from structurally and rhetorically represented technical documents. The availability of these Core Documentation Entities leverages the capabilities of semantic information systems, as information can be accessed on a higher detail level (e.g. repair procedure instead of chapter).

In summary, the first research gap that is tackled by this thesis is the fact that existing vocabularies for semantic publishing do not fit the domain of technical documents. Additionally most of these vocabularies do not max out the capabilities of semantic representations. Therefore, this work adapts the idea of describing structural and rhetorical document components to the special features of technical documents. Hereby, the approach carefully considers existing industry standards to facilitate an easy migration. Besides the easy migration and integration another important aspect is to enhance the vocabulary with logical expressions that allow the derivation of Core Documentation Entities from structural and rhetorical descriptions.

### 1.2.2 5-STAR Semantification

A semantic representation of technical documents (Linkable Technical Documents) as described before is a fundamental requirement for semantic information retrieval. However, the incorporation of these technologies in enterprise applications has just started and thus only a small amount of newly created technical documentation is already semantically enriched. However, large corpora of legacy technical documents



without meta-data (1 star data) exist. As corresponding machines often have product life cycles of several decades these corpora also need to be integrated in new semantic information systems.

The integration requires the creation of links between the semantic representation of a technical document and corresponding concepts from the Linked Enterprise Data graph that describe its topics. Therefore, the legacy documents first need to be transformed automatically to a semantic representation. Then, for establishing the link between the semantic representation of the document and concepts in the Linked Enterprise Data graph, the text needs to be thoroughly analyzed in order to identify its main topics.

A holistic approach for the described transformation of legacy technical documents (1 star) to a linked semantic representation (5 stars) does not exist. Existing methods tackle subtasks, but do not consider special features of technical documents. Additionally, most methods lack a comprehensive methodology that allows to measure the respective performance.

Hence, the second part of the thesis concentrates on the transformation of legacy technical documents (1 star) to Linkable Technical Documents (5 star). The thesis will outline a journey that describes in detail how each maturity level (star) can be achieved; what benefits arise and what efforts need to be undertaken (cost-benefit ratio). As described before the underlying transformation process comprises several tasks that build upon each other and are closely aligned with the 5-STAR maturity schema:

**1. Document Layout Analysis:**

The 5-STAR maturity schema requires 1-STAR technical documents to be available in an electronic format that gives access to high-level structures like pages, blocks, texts, and tokens. This requirement is mapped to the more general problem of *Document Layout Analysis*.

**2. Logical Document Structure Recovery:**

The second maturity level requires documents to provide access to typed document structures like paragraphs, headlines, lists, and tables. This problem is mapped to the more general problem of *Logical Document Structure Recovery* that also covers additional aspects like the determination of the reading order.

**3. Modularization:**

The goal of the 3-STAR semantification is the subsequent recovery of a document's chapter structure with the goal of deriving self-contained modules (information units). The modularization problem is divided into three tasks: (1) reconstructing the chapter structure of a document on the basis of 2-STAR information, (2) identifying duplicate content and (3) aligning modules that exist in multiple languages.

**4. Identifiability and Information Types:**

The fourth maturity level requires identifiability and information typing from technical documents. The problem of identifiability is motivated by the desired integration in an Linked Enterprise Data graph and is approached with persistent uniform resource locators (PURL). The information typing aims on specific Linked Data applications, e.g. the rhetorical filtering of modules with respect to an information type. The problem of assigning information types to modules is mapped to established methods from *Automatic Document Classification*.

### 5. Subject Analysis:

The 5-STAR semantification aims at the identification and annotation of the main subjects for a given technical document. This is related to the more general problem of *Subject Analysis* or *Subject Indexing*. In the context of this work, subjects are nodes of a Linked Enterprise Data graph. Annotating documents with such nodes actually establishes a link between the document and the Linked Enterprise Data graph and thus fully integrates the document into the Linked Enterprise Data.

All of these subtasks are covered in detail; existing methods get discussed and evaluated. Novel and tailored solutions that are adapted to the special features of technical documentation are introduced where necessary. Additionally, practical recommendations and suitable formats are provided for each semantification step.

### 1.2.3 Domain Knowledge as the Regulating Screw

The work described in this thesis is in large parts based on knowledge-based methods. Although a lot of work in the fields of Machine Learning for Natural Language Processing tasks has been published, it can practically not be applied to the problem of semantifying technical documents. The main reason is the absence of respective training data. In real-world projects the creation of training data is usually difficult to achieve, as (1) required experts are bound in important projects and (2) the resulting performance of a Machine Learning system is difficult to estimate. In contrast, knowledge-based systems usually allow for an iteratively incremental project implementation. First results are usually available in early project phases and can be coupled to decision-points and milestones. More importantly, technical documentation often follows strict rules that regard structuring, formatting and other aspects. These rules can be exploited by knowledge-based methods and leverage their performance. This makes domain knowledge an important regulating screw for the presented semantification approach.

### 1.2.4 Unified Pipelines and Integrated Environments

An important success factor for the presented approach is its broad applicability and scalability. Applicability and scalability concerns human users and the underlying data. This aims on providing software tools that are applicable to a lot of different data sets with minimum adaptation efforts. Thus, this thesis introduces a semantification architecture that aims on maximum reuse of developed components by providing a large library of semantification components and unified pipelines for specific formats. The semantification architecture is accompanied by an integrated semantification environment that aims on enabling users with different skill levels to perform semantification tasks. Therefore, a Graphical User Interface provides easy access to predefined pipelines and allows for easily configuring single processing steps. Advanced users can extend the functionality of the semantification architecture with their own plugins that encapsulate tailored semantification steps. Therefore, the semantification architecture is based on a plugin mechanism and a standardized but extensible data model.

## 1.3 Results

The presented work describes the semantification of technical documentation for their integration in Linked Enterprise Data graphs. The semantification comprises both

the identification of a suitable semantic representation under consideration of existing semantic publishing approaches and (de-facto) standards for modeling technical information as well as a holistic approach that transforms existing data into such a representation. Thus, this work contributes a generally applicable approach for the integration of legacy technical documentation into state-of-the-art information systems.

The semantic representation combines structural and rhetorical aspects of technical documents and provides logical expressions that allow to identify and access structures that carry strong technical knowledge — Core Documentation Entities. However, the identification of Core Documentation Entities requires a certain maturity of technical documents, i.e. the availability of structural and rhetorical information. The idea of measuring the maturity of technical documents with respect to accessibility of such knowledge structures has been presented in Furth et al. [64].

The availability of Core Documentation Entities dramatically increases the accessibility of technical documents. This is beneficial for human users in a couple of information retrieval scenarios. The increased accessibility, additionally, enables new application scenarios for legacy technical documents. This comprises the targeted extraction of information from technical documents for Ontology Learning and Ontology Population purposes. The construction of technical documents from document structures (Core Documentation Entities) is described in Furth et al. [63]. An accompanying methodology for developing ontologies from document structures has been described in Furth et al. [66]. Resulting ontologies are usually rather large and may contain inconsistencies. Thus, the resulting ontologies might require manual post-processing. A delta-debugging algorithm for ontologies that improves this task is presented in Furth et al. [62].

Although a variety of application scenarios exists for semantically described technical documents only small portions of existing corpora are actually semantically prepared. The holistic approach described in this thesis consists of five consecutive steps that support the transformation of documents in legacy formats (scanned images or PDFs) to the described semantic representation. The application of an early and preliminary version of this process to large amounts of technical documents is described in Furth et al. [65].

The process requires the recovery of document structures on different granularity levels. Hereby, a fundamental step is the classification of single blocks with respect to their original type. The knowledge-based recovery of block-level elements like headlines, paragraphs, lists, and tables is described in Furth et al. [70]. The final step of the process links the recovered structures to concepts from a Linked Enterprise Data graph. Therefore, a thorough analysis of the corresponding text is necessary in order to identify the key subjects. A series of Subject Analysis and Subject Indexing approaches has been described in prior work [67, 65, 68].

The different steps of the semantification are supported by an integrated semantification architecture. Requirements, architecture, and implementation remarks are presented in Furth et al. [69]. The architecture also allows to integrate visual review components. A visualization using the city metaphor is presented in Baumeister et al. [13].

The semantification process is accompanied by a tool box of semantification applications. The tools support different steps of the semantification process ranging from the knowledge acquisition for the classification of document structures over an integrated semantification architecture that enables the batch processing of large corpora to dedicated review tools. The tools demonstrate the practical applicability of the semantic representation and the accompanying semantification approach.

The semantic representation, the semantification process and the accompanying tools have been successfully applied in a series of industrial projects. This thesis reports on a series of real-life semantification projects for different customers within the domains of harvesting technology, earth moving technologies and special purpose vehicles. All projects aimed on semantifying legacy documents for their subsequent usage in modern information retrieval systems. The presented approach also supports the integration of technical documentation in other applications that are in frequent use in daily maintenance and repair task. Thus, this thesis also reports on the creation of a unified pipeline that enables the linking of technical documents with electronic spare parts catalogues.

## 1.4 Structure of this Work

This work is structured into seven chapters. The first chapter, concluding with this outline, gave an overview of the research goal, the approach and results. The remainder of this work is organized as follows:

- **Chapter 2** introduces technical documents as a special knowledge resource and discusses existing challenges regarding accessing and linking them. Therefore, a brief overview of relevant semantic technologies is given within the context of Linked (Enterprise) Data. The chapter also presents basic text analytics methodologies that are required for the integration of technical documents into a Linked (Enterprise) Data graph. The chapter concludes with a presentation of Semantic Information Retrieval, which is one of the most recognized applications of Linked (Enterprise) Data.
- **Chapter 3** starts with the presentation of a novel maturity model that reflects the accessibility and linkability of technical documents. Building upon this maturity model an abstracting semantic meta representation for technical documents is introduced that gives access to encapsulated fine-grained technical knowledge in written texts. This meta representation is then mapped to existing information models for technical documentation.
- **Chapter 4** challenges the so called *legacy gap*, i.e. the fact that large amounts of technical documents exist in formats that are not semantically prepared. Therefore, a novel and holistic process for the semantification of technical documentation is presented. The process consists of five consecutive steps that are closely aligned with the maturity model. For each step a thorough presentation of existing and novel approaches as well as practical recommendations are given.
- **Chapter 5** gives a brief overview of the reference implementation of the semantification process. Therefore, an extensible semantification architecture is described. The architecture is accompanied by an integrated semantification environment.
- **Chapter 6** explains the semantification process and the usage of the developed semantification environment. Therefore, the complete 5-STAR semantification of an example document is described.
- **Chapter 7** presents several practical applications of the holistic semantification process. The focus is on several industrial case studies that underline the practical applicability of the approach.

- **Chapter 8** summarizes the work described in this thesis. Additionally, future research questions with respect to the semantification of technical documents are discussed. The chapter also provides an outlook to the applicability of the approach in other problem domains.



## Chapter 2

# Accessing Technical Documents

*Lack of documentation is becoming a problem for acceptance.*

---

*Wietse Venema*

## 2.1 Technical Documentation<sup>1</sup>

Builders of machinery and plants provide technical documentation to support the service technician to ensure the safe operation and maintenance of their products. In the past, the documentation was printed on paper. With the increasing complexity of the machines many vendors switched to electronic versions of the books in recent years (PDF and HTML). For instance, the documentation for a full-featured harvesting machine or other special purpose vehicles comprises about 10,000 pages. With the electronic availability the metaphor of a single 'book' is not necessary anymore.

### 2.1.1 Characteristics of Technical Documentation

In this section, the domain of technical documentation is introduced in more detail. The understanding of the different types and uses of the technical documentation is helpful to fully understand the motivation of the later introduced semantic technologies.

#### Tasks of the Technical Documentation

The main task of technical documentation is the support and training of a service technician during daily work. Here, the documentation is used to teach entire functional systems but also to fill knowledge gaps during troubleshooting and maintenance of the technician on-site. Therefore, technical documentation needs to work as a teaching textbook but also as a lexicon. The service technician is supported by the documentation during the following tasks:

1. Operation of the machine
2. Maintenance of the machine
3. Diagnosis of problems
4. Repair of damage

### 2.1.2 Types of Technical Documentation

The described tasks are supported by documentation manuals. The organization of the manual into one large file or into multiple documents varies from company to company. For larger and more complex machines companies tend to organize the technical documentation into multiple documents, each covering a specific aspect. The main aspects and their corresponding manuals are introduced in the following.

---

<sup>1</sup>The contents of this section have already appeared in a slightly different version in the following published article: Sebastian Furth, and Joachim Baumeister. "Semantification of Large Corpora of Technical Documentation." Enterprise Big Data Engineering, Analytics, and Management, IGI Global, 2016 [65].



### **Operation Manual**

The operation of each machine function is described in detail for a non-technical user. For instance, for driving machines the use of the "gear-stick" is described, i.e., how to select an appropriate gear for the current driving speed. Also, simple and frequent maintenance tasks accomplishable by end-users are explained.

### **Repair Manual**

The repair manual targets the technical user, typically service technicians, and it describes the repair of all relevant components of the machines. In such a manual, the exchange and adjustments of mechanical parts is explained. Following the example, the repair manual would describe the replacement of a defective "gear-stick". Depending on the manufacturer of the machine, different levels of detail are used in the manual, ranging from the description of the exchange of only larger components to the repair of elements of detailed sub-components. For electronic parts and software, the update, and calibration of the particular entities are described.

### **Diagnosis Book**

For the diagnosis of malfunctions, the service technician needs to have a thorough understanding of the functional dependencies and interrelations of the components. The diagnosis book describes for each technical function the connections between components. For instance, for a driving machine the components "gear stick", "CAN bus", and "transmission" are connected in a functional dependency. Typically, also the electrics and hydraulics of a machine are documented in such a manual, e.g., by including circuit diagrams.

### **Spare Parts Catalog**

The spare parts catalogs provide a detailed view of the parts located in particular components. Service technicians use such catalogs to locate specific parts, but also to order new parts in exchange for defective ones. Typically, the catalogs are defined hierarchically, starting from top-level components (e.g., a "cabin") and then navigating to detailed components contained in the top-level components (e.g., a "gear-stick" in a "cabin").

## 2.2 Semantic Technologies

This chapter describes selected semantic technologies. The *Semantic Web* and the underlying semantic technologies have become broad research fields that provide solutions for a plethora of problem scenarios. Therefore, the following sections concentrate on methods and technologies that are considered to be relevant for realizing the vision of linkable and accessible technical documents.

### 2.2.1 In a Nutshell

Berners-Lee et al. [18, 19] proposed the semantic web as a composed architecture of standardized technologies. The accompanying *Semantic Web Stack* (see Figure 2.1) shows how the different technologies are organized for the realization of semantic web solutions. Technologies on higher levels can use functions and features of lower levels. The stack gets continuously updated and is thus intended to reflect the current implementation progress of semantic technologies. At the time of writing, technologies up to OWL have been implemented and standardized. The levels above have not yet been implemented or standardized, though a couple of approaches exist.

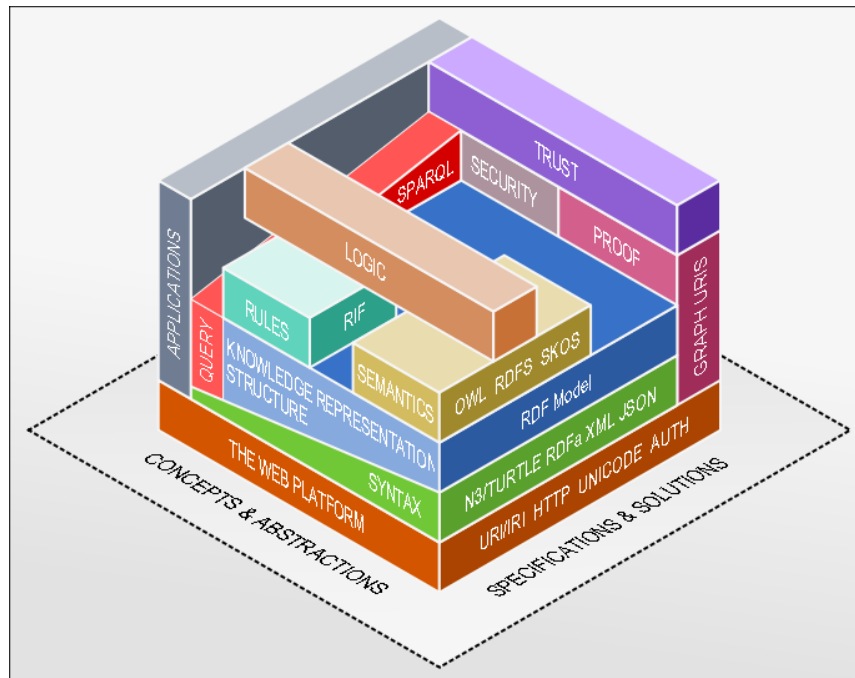


FIGURE 2.1 | “Visualization of the common, layered Semantic Web technology stack” by jsaiya / CC BY 3.0.

The lowest levels of the semantic web stack describe basic technologies that have been successfully employed for many years in the traditional web. The *Uniform Resource Identifier* (URI) [17] or its generalized form *Internationalized Resource Identifier* (IRI) [55] are prominent examples. They are used to identify resources in the (semantic) web. *Unicode* is another fundamental technology that is broadly used for the representation of texts in different languages.

The next layer is concerned with syntax. Here prominent languages like the *Extensible Markup Language* (XML) [26], *Turtle* [15] or *JSON* [47]/*JSON-LD* [186] reside. These languages support the usage of different namespaces, which allows the usage and linking of markups from multiple sources. This feature is the basis for

the next level of the technology stack, which is concerned with knowledge representation structures. The *Resource Description Framework* (RDF) [107] was the first technology to be standardized by the World Wide Web Consortium especially for the semantic web. The basic idea is to describe asserted statements as *triples*, which get interconnected in a graph.

Different semantic vocabularies build on top of the RDF language features. The ontology language *RDF Schema* (RDFS) [30] is primarily a vocabulary collection for RDF. The vocabulary provides language features for the definition of class or property hierarchies. The *Web Ontology Language* (OWL) [14, 110] can be considered as an extension of RDFS, which provides more sophisticated language constructs for the semantic enrichment of RDF data. For the RDFS and OWL vocabularies reasoning logic exists that can be employed to infer additional triples from given data. SPARQL [83] is a query language for data in RDF format. The Rule Interchange Format (RIF) [23] is intended to support the exchange of rules originally defined in different rule languages. The *Semantic Web Rule Language* (SWRL) [92] is a rule language that allows the definition of rules for OWL ontologies. However, SWRL has not yet been standardized.

### 2.2.2 Types of Ontologies

Ontologies are intended to describe different types of entities of the real world [88]. This is especially true in the context of the semantic web, where special vocabularies for the description of ontologies has been standardized, cf. RDF Schema (RDFS) [30] and OWL [14, 110]. Ontologies fulfill different purposes and there is no commonly agreed upon classification of respective ontology types. This thesis uses the widely accepted classification scheme that has been proposed by Heijst et al. [86, 85]. The classification scheme mainly considers the subject of conceptualization for the definition of ontology types. Relevant types of ontologies according to this classification schema are:

- **Meta ontologies:**

The most common vocabulary that is usually valid for multiple problem domains is defined in meta ontologies. Hence, meta ontologies contain general classes and properties that are later instantiated for concrete problems. In the context of technical (document) ontologies a meta ontology might provide vocabulary for the definition of machines, documents and more fine grained document structures and components.

- **Domain ontologies:**

The term of domain ontologies is not defined precisely. In the context of this work domain ontologies are seen to contain instantiated knowledge on basis of meta ontology classes. In the context of technical (document) ontologies a domain ontology might describe concrete machinery or its corresponding corpus of documents. A special characteristic of technical documents is the usage of a relatively fixed and sometimes controlled vocabulary that is closely related to the machinery in focus. This characteristic can be exploited when technical documents and concepts from technical domain ontologies get interlinked. Then, the set of concepts that shall serve as document annotations can easily be derived from the structural description of a machine that is contained in a technical domain ontology.

- **Application ontologies:**

Knowledge that is required for specific applications gets modeled in application

ontologies. Application ontologies usually refer to domain ontologies, sometimes they are closely coupled or even included in domain ontologies. In the context of linkable technical documentation an application ontology might describe knowledge that is required for semantic information retrieval. Therefore, semantic indexing information like chapter titles, page ranges or relevant subjects might be described in a designated application ontology.

### 2.2.3 Linked Data

In recent years the realization of Linked Data [19] is one of the most recognized results produced by semantic technologies. Linked Data extends the traditional internet, which mainly consists of websites, by a net of data. The main goal of the Linked Data initiative is to connect heterogeneous data from different sources in order to gain new insights. Therefore, data on the web needs to be described using the aforementioned semantic technologies. The main requirements of such data is its ability to be referenced and unambiguously linked, i.e. it needs to be reachable via URIs/IRIs. Additionally, resources usually need to be enhanced with metadata in order to ensure machine readability.

Linked Data exists in different forms ranging from data published by public administrations to data that follows the idea of *Linked Open Data* (LOD). Linked Open Data is freely available on the internet, can be identified by URIs or IRIs and links itself to other public data sets. The World Wide Web Consortium (W3C) described guidelines for Linked Open Data that aim for the easy usage of such data, especially regarding machine readability. The combination of different data sets across domains and organizations usually yields new insights. The technical basis of Linked Open Data is the Resource Description Framework (RDF) [107]. Existing structured data sources can usually easily be transformed to the RDF format. The transformation to RDF is usually challenging for complex unstructured data – like technical documents.

Several large, interconnected and free data sets with billions of facts and connections have been built in the last years. The Linked Open Data Cloud in Figure 2.2 visualizes the contents of the most popular data sets like DBPedia<sup>2</sup>, Open Street Map<sup>3</sup>, and Linked Geo Data<sup>4</sup>.

### 2.2.4 Linked Enterprise Data

Structured and unstructured enterprise data has traditionally been organized in relational databases and document management systems respectively [21]. Structured company master data is managed and defined according to strictly defined database and metadata schemes [21]. In contrast, unstructured information like written texts usually have only little or no explicitly defined schemes but follow the rules of natural language [21]. In consequence, the interconnection of structured master data and unstructured texts is often hardly possible in traditional IT landscapes.

The idea of Linked Enterprise Data is to interpret, process, interconnect, and adequately present enterprise data and information from different sources and structures, such that efforts for information acquisition and retrieval decrease significantly [21]. Therefore, Linked Enterprise Data employs semantic technologies like the Resource Description Framework (RDF) [107] for building a data integration layer for different structured and unstructured information [21]. Therefore, controlled vocabularies

---

<sup>2</sup><http://dbpedia.org>

<sup>3</sup><http://www.openstreetmap.org>

<sup>4</sup><http://linkedgeodata.org>

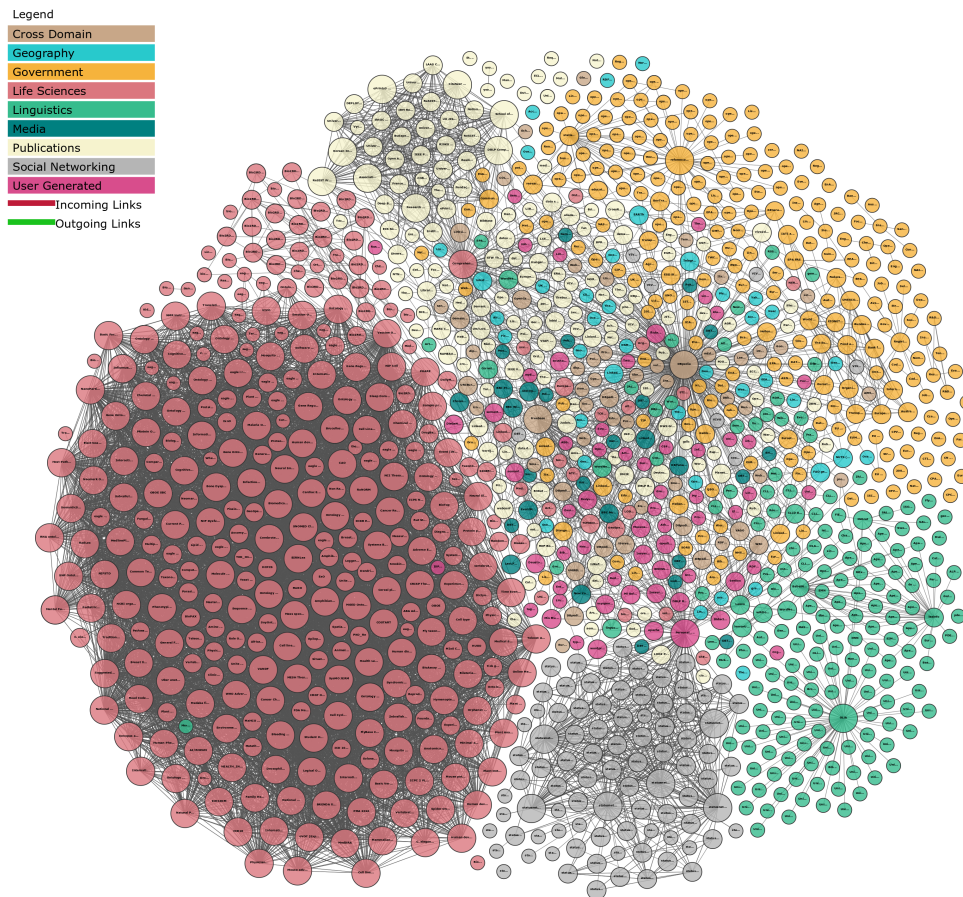


FIGURE 2.2 | “Linking Open Data cloud diagram 2017”, by Andrejs Abele, John P. McCrae, Paul Buitelaar, Anja Jentzsch and Richard Cyganiak. (<http://lod-cloud.net>).

organized in enterprise knowledge graphs / ontologies are used to map syntax and semantics of relevant information [21]. Blumauer [21] claims that using semantic technologies instead of traditional XML based mechanisms yields the following advantages:

- Linked Data based data models are less abstract than XML schemes or relational databases. [21]
- Linked Data based data models connect information for human beings and machines in one model. [21]
- Knowledge Graphs / Ontologies can be developed incrementally and evolve with new requirements. [21]
- Linked Data allows to semantically represent and interconnect structured and unstructured data. [21]
- Linked Data has a powerful query language (SPARQL). [21]
- Linked Data Graphs can directly improve user experience. [21]

Based on these advantages Blumauer [21] identifies three main application scenarios for Linked Enterprise Data.

- **Data Integration with Linked Data:**

Linked Data principles and semantic technologies are used internally for data integration purposes and semantic search. This usually supports all business processes and units that work with complex, heterogeneous and distributed data as time required for information retrieval usually decreases significantly. [21]

- **Integrating Data from the Linked Data Cloud:**

Other enterprises also consume data from the Linked Data Cloud in order to enhance internal applications and databases. Prominent examples are the addition of geo data to e-mails and tickets or the highlighting of words in written texts with explanatory semantic links. [21]

- **Publishing Data to the Linked Data Cloud:**

Some enterprises also publish their own data to the public Linked Data Cloud. The main goal is usually to develop new markets for distribution. The publishing of data to the Linked Data Cloud is usually accompanied by a combination of open and closed license models. This is especially true for media companies and book publishers. [21]

## 2.3 Text Analytics for Linked Data Integration

### 2.3.1 Linked Data Lifecycle

Auer et al. [4] confirm that data is becoming more and more a success factor for existing enterprises. Therefore, they claim that the lifecycle of data in enterprises must be comprehensively supported [4]. This is especially true for data that is integrated in a company's Linked Enterprise Data graph. They propose the Linked Data Lifecycle (see Figure 2.3) that describes especially the following phases for Linked Data in enterprises:

- **Storage / Querying:**

Linked Enterprise Data is represented as RDF. As RDF statements are essentially expressed as triples they are usually stored in respective triple stores. Additionally traditional databases with RDF interfaces might be employed. Enterprises are usually confronted with additional challenges that concern storage and querying like caching, query optimization or scalability. [4]

- **Manual Revision / Authoring:**

A vital element of Linked Enterprise Data is the semantic authoring of new data. Therefore, adequately equipped authoring tools and editors should encourage the semantic enhancement of all kinds of data. [4]

- **Interlinking / Fusion:**

The automatic interlinking of single data sets to a complete Linked Enterprise Data graph is a huge challenge in most companies. This is especially true for unstructured data where unlike in traditional databases the simple mapping of keys is usually not possible.

- **Classification / Enrichment:**

All data that shall be included in a Linked Enterprise Data graph needs to be mapped to respective taxonomies and vocabularies from enterprise ontologies. This mapping is especially challenging for unstructured data like written texts. [4]

- **Quality Analysis:**

The quality of internal and external data usually varies dramatically. Therefore, methods for the automatic evaluation of data quality are required. Prominent measurements are based on data provenance, context, coverage, and structure. [4]

- **Evolution / Repair:**

Data is typically dynamic and evolves over time. It is critical for Linked Enterprise Data that evolving data remain stable and consistent. Therefore, changes to vocabularies and ontologies must be transparent. Automatic test methods like continuous integration for ontologies or ontology debugging support the discovery of inconsistencies [9] and their repair. Previous work describes the delta debugging of ontologies [62] that supports the repair task. [4]

- **Search / Browsing / Exploration:**

Users do typically not yet have a high affinity to linked data and the interaction with it. Therefore, relations between data should be visible in intranet and internet applications and search, browsing, exploration and visualization methods should support the usage. [4]



FIGURE 2.3 | “Linked Data Lifecycle”, by Auer et al. [4].

### 2.3.2 Linked Data Integration in a Nutshell

The Linked Data Lifecycle that has been described in the previous sections is the basis for the Linked Data Integration task. The Linked Data Integration task is especially concerned with the Interlinking / Fusion and Classification / Enrichment phases of the Linked Data Lifecycle. Isele [98] describes Linked Data Integration as a process with three main steps:

- **Data Translation:**

The data translation sub task of Linked Data Integration is concerned with the translation of existing data sets to an RDF schema that fits an enterprises’ needs. It is usually necessary to distinguish the handling of internal and external data as well as data that already exists in RDF format and other data. Linked Data mappings are usually employed for the translation of existing structured data. The translation of unstructured data, however, is usually more challenging and will be described for technical documents throughout this thesis. [98]

- **Entity Matching:**

When all data has been converted to an RDF representation the entities of all datasets must be mapped to each other. This mapping process is usually referred to as Entity Matching and tries to identify duplicate entities in different data sets. Different methods exist for the entity matching problem ranging from automatic approaches that consider domain-independent background knowledge to rule-based approaches. Rule-based approaches employ domain-specific rules that determine the similarity between entities of different sources. Distance metrics for strings are a vital foundation for these rules that must compare different textual attributes of the respective entities. Then, automatic classifiers are employed to compute entity mappings on the basis of the computed similarities of their respective attributes. [98]



- **Data Fusion:**

The data fusion step tries to unify already mapped entities, i.e. it determines an entity representation that reflects the information from all mapped instances. Important aspects of the fusion are completeness and consistency. While completeness is usually assured by simply considering all links of an entity consistency is a bigger issue. Ensuring consistency might be challenging when data sources contain different values for specific attributes of an entity. [98]

### 2.3.3 Text Analytics as Part of Linked Data Integration

Text Analytics is an established research field and provides a variety of approaches that can be exploited for the Linked Data Integration task for unstructured information in the form of written text. This section gives a brief overview of basic text analytics techniques that support the data translation step for unstructured textual information that is part of the Linked Data Integration task.

#### Preprocessing

In general, the Data Translation (see Section 2.3.2) of textual information into an RDF representation first requires a series of common preprocessing steps:

1. **Tokenization:**

The most elementary preprocessing step is the detection of sentence and word boundaries. This is usually referred to as Tokenization (or sometimes Zoning). Seldomly, tokenization is also considered to be the process of detecting paragraphs, sections, chapters, and more generally the detection of reasonable document components. In this thesis tokenization is considered as the process of detecting words and components. The definition of a *graphic word* of Kucera et al. [111] is a good basis for the detection of word boundaries: “a string of contiguous alphanumeric characters with space on either side; may include hyphens and apostrophes, but no other punctuation marks” [111]. Practical applications, however, usually include data where the word detection on the basis of the aforementioned definition fails.

Manning and Schütze [126] thoroughly discuss a series of challenges concerning the detection of word boundaries. In the following paragraph, a selection of these challenges is presented. According to the graphic word definition words can be identified by checking their limitation with whitespace characters. However, this is especially problematic for abbreviations like “etc.”. Simply ignoring the dot as last character would at least fail when the abbreviation appears at the end of sentence, where the dot character then fulfills two functions. This phenomenon is referred to as *haplogy*. According to Kucera et al. [111] single quotes are part of a word. In the English language, however, this consideration is problematic. Cases like “I’ll” or “isn’t” illustrate that considering the single quote as part of a word would produce an error, as the presented examples obviously consist of two words. A similar phenomenon arises from the usage of hyphens in words like “co-operate”. In this case it is, however, not clear whether the respective words should be considered separately. The often inconsistent usage of hyphens must also be considered. In practice, however, whitespace tokenization approaches are very popular although they have to deal with the aforementioned challenges. Respective tokenization results are usually good to very good for English texts. Other languages introduce additional challenges to the tokenization problem. A popular example from the German language

is the compound of words like "Öldruckschalter". The decomposition of the word into its compounds would often be beneficial but is a practically unsolved problem, especially for domain-specific (technical) terms. Other languages like Chinese or Japanese do not even make use of whitespace characters which makes the tokenization with the presented approach impossible. Another problem is the existence of different spellings for a word like "database", "data-base" or "data base". In this case a separation on the whitespace character should be suppressed. This is also true for strings that follow special patterns like phone numbers. The detection of sentence boundaries is usually based on characters like ".", "?" or "!". However, the handling of characters like ":" are ";" is often not clearly defined. Manning and Schütze [126] claim that most of the described challenges can be handled by using respective heuristics.

## 2. Morphological and Lexical Analysis:

After sentences and words have been detected through tokenization approaches a morphological and lexical analysis follows. Amongst others, Feldman et al. [58] and Manning et al. [126] thoroughly describe these preprocessing steps. The morphological and lexical analysis usually consists of two main components: the Part-of-Speech-Tagging and the Sense Disambiguation. The latter is concerned with finding the correct lexeme for homographs. An example is the English word "saw", which could either be the past tense of the word "to see" or the noun "tool". The Part-of-Speech-Tagging on the other hand assigns the single words of a sentence tags according to their role. The tags are not standardized, however usually they refer to the following seven base categories: articles, nouns, verbs, adjectives, prepositions, numbers, and named entities. Additional tag sets exist that are more extensive. Examples comprise the Brown Tag Set<sup>5</sup>, the C5 Tag Set<sup>6</sup> and the Penn Treebank Tag Set<sup>7</sup>.

## 3. Syntactic Analysis:

The syntactic analysis is usually concerned with parsing the grammar of sentences. A thorough description of this task is given by Feldman et al. [58] or Manning et al. [126]. A sentence can, for example, be analyzed according to a Probabilistic Context Free Grammar (PCFG) [39]. Basically, two grammar concepts are distinguished. The first and rarely applied form of grammar is the Dependency Grammar. Here, sentences get parsed in order to identify dependencies between single words. The other concept describes Constituency Grammars, where a sentence is split into its compounds: noun phrase, verb phrase, prepositional phrase, adjective phrase and clause. The phrases usually get roles assigned like subject or object. With respect to their efficiency both forms of grammars are usually too expensive for practical usage. Therefore, faster and more robust Shallow Parsing [178] approaches are applied. Shallow Parsing approaches do not require a full parse of a sentence and instead simply annotate the simple and unambiguous parts of sentences.

## 4. Anaphora / Coreference Resolution:

The resolution of anaphoras (coreferences) follows the syntactical analysis and aims at identifying phrases that refer to the same entity. Anaphoras appear in different forms. Feldman et al. [58] give a good overview. The most recognized form of anaphora is the pronominal anaphora, which result from the

---

<sup>5</sup><http://www.comp.leeds.ac.uk/ccalas/tagsets/brown.html>

<sup>6</sup><http://www.natcorp.ox.ac.uk/docs/c5spec.html>

<sup>7</sup><http://www.comp.leeds.ac.uk/ccalas/tagsets/upenn.html>

usage of pronouns like "he" or "she". Due to its frequent appearance and its comparably simple handling this form usually is in focus. Approaches are either knowledge-based or use machine learning methods. The latter require respective training data, while knowledge-based approaches are usually specially customized algorithms. Established knowledge-based approaches are the algorithm of Hobbs [90], CogNIAC [6] and the approaches of Kennedy et al. [104] and Mitkov [141]. An example for a machine learning approach is the method proposed by Soon et al. [184].

The basic preprocessing of textual information is similar and independent from the actual use case. However, depending on the application scenario and the respective target RDF representation, the further processing is different. For the Data Translation task of the Linked Data Integration, two application scenarios are essential: Entity Recognition and Document Classification/Subject Analysis. Both application scenarios aim on translating unstructured textual data into a representation that can be connected to an existing Linked (Enterprise) Data graph. However, the resulting links of the different approaches differ significantly. While Entity Recognition typically results in a bag of entities that can all be used for establishing links to the Linked (Enterprise) Data graph, the translation using Document Classification/Subject Analysis approaches aim on significantly reducing the amount of links. Instead of establishing connections to all discovered entities, a document gets classified according to predefined classes/subjects that are chosen from an existing Linked (Enterprise) Data graph. The next section gives a brief overview of the Entity Recognition problem and discusses the usage of different technical term sources. Section 2.3.3 shortly points out some details about document classification for the Linked Data Integration task.

### Entity Recognition

The Entity Recognition task, as described amongst others by Nadeau [143] or Feldman et al. [58] is concerned with detecting all proper nouns and units in a text. In normal texts such entities typically comprise names of persons, locations or organizations, information about date and time, amounts of money or percent values. In the context of technical documents, relevant entities typically comprise names of machines, machine variants, components, functions, measurement values, filling quantities, and maintenance intervals. Then, having all entities appearing in a text available they all might be simply linked to the respective concepts of a Linked (Enterprise) Data graph. Therefore, Entity Matching approaches (see Section 2.3.2) must be applied.

### Overview

Detecting known entities is usually a rather trivial task under consideration of respective dictionaries. A more challenging subtask is the detection of unknown entities. The detection of such unknown entities usually requires specialized methods that have been mainly based on handcrafted rules in the past. Recently, approaches based on Machine Learning gain importance. Hence, the currently predominant form of Named Entity Recognition is based on supervised Machine Learning methods. Respective results are usually promising but require training data. Due to the usually missing training data with respect to technical documents, the respective approaches are not discussed in this thesis. Nadeau [143] surveyed the topic, discusses different approaches and points to methods that cope with small amounts of training data.

Nadeau [143] also recommends the usage of combined rule-based and dictionary-based approaches under absence of respective training data. The required dictionaries can have different sources and typically comprise different kinds of lists, like lists of entities or lists of keywords. Then, using the dictionary information, all occurrences of entities or keywords in texts are identified. Especially on the basis of rules operating on keywords and patterns, additional entities can be detected. An example for such an approach was proposed by Sekine et al. [177]. Entity Recognition approaches that are based on dictionaries and rules are, however, often closely coupled to the respective domain. The usage of such systems on more general texts typically yields rather poor outcomes like the evaluation of the approach of Sekine et al. [177] confirms.

### Sources of Technical Entities

The fundamental requirement for the application of dictionary-based Entity Recognition approaches is the availability of terms that represent the respective entities. In the mechanical engineering domain, typical examples for term sources are stock lists or technical structure models. Figure 2.4 shows an excerpt of a stock list, that contains terms (highlighted) and a lot of additional information (e.g. identifiers, amounts etc.) about the parts of a specific machine. Another valuable terminology source is

Catalogue sequence number	Item sequence number	Reason for selection	Quantity per next higher assembly	Manufacturer	Part number	2000M PAS Segment						
						Description of parts	Unit of issue	2000M PCS Segment	Special storage	Fitment code	Physical security/pilferage code	Calibration marker
B5000001 064	00A	1	1	A434	1954343534079	BEAR SELECTOR SWITCH	EA		0		U	
B5000001 065	00A	1	6	99008	040328A2P	NUT,PLAIN,HEXAGON	EA		0		U	
B5000001 066	00A	1	6	99008	7089200HV	WASHER,FLAT	EA		0		U	
B5000001 067	00A	1	3	11871	2700109-060000.007.0	MOUNT,RESILIENT,GENERAL PURPOSE	EA		0		U	
B5000001 068	00A	1	1	33518	278M531SCHW	KNOB	EA		0		U	

FIGURE 2.4 | Excerpt of a stock list showing the contained terms.

the corpus itself. In recent years controlled vocabularies were introduced for the authoring process. However, older documents often use varying vocabularies because of different writing styles or translation processes. Therefore especially the older technical documents in the corpus are a promising source for the extraction of terms in a preceding step. This preceding step uses Automatic Term Extraction (ATE) methods that can be used to extract term candidates from textual resources automatically. Popular examples are the C-Value/NC-Value method [61], TermEx [175] and GlossEx [153], that combine linguistic and statistical measures to extract ranked lists of terms. For further reading at this point please refer to the evaluations of ATE methods from Pazienza et al. [155] and Zhang et al. [200].

### Quality of Technical Terms

The term sources usually vary in term quality. While a high quality extraction of terms from structured sources can be assumed, the terms extracted using ATE methods need a dedicated quality measure. Thus the quality of these terms is usually expressed as a confidence value computed by an ATE method. These confidence values typically consider the termhood (degree to which a linguistic unit is related to domain-specific concepts) and unithood (strength and stability of word combinations or collocations) of a term, as defined by Kageura et al. [102].

A large terminology is crucial for the performance of dictionary-based entity recognition. In general, the more terminology is available, the better the entity recognition is. However some term sources contribute more to the performance than others, as

they better reflect the terminology used in the documents. Additionally, low quality terms might result in bad Linked Data Integration results in form of wrong links. Hence, a good strategy is to prioritize term sources during the integration process. Albeit the prioritization of term sources according to their contribution capabilities is not a trivial task. Usually even technical writers and editors have to guess, when they are asked to prioritize term sources according to the expected compliance with a corpus. Objective metrics can support the technical editors in their cognitive selection process. An example for such a metric is the *term coverage*, which expresses the proportion of terms among all tokens in a text.

Therefore, a *token* is formally defined as follows:

**Definition 2.3.1** (Token). Let  $K$  be the universal set of all documents (corpus) and let  $L$  be the universal set of all possible literals. Then the function  $to : K \rightarrow 2^L$  extracts the set of all literals for a given document. We call the literal  $t \in to(D_i)$  a *token* of document  $D_i$ , where  $D_i \in K$ .

Additionally, *term candidates* are defined as follows:

**Definition 2.3.2** (Term Candidates). Let  $L$  be the universal set of all possible literals. Then we define the *term candidates*  $T'$  of a problem domain as a set of literals, i.e.,  $T' \subseteq L$ .

Having formal definitions of terms and term candidates available the term coverage is defined as follows:

**Definition 2.3.3** (Term Coverage). Let  $T' \subseteq L$  be the term candidates of the problem domain extracted from a term source, and let  $K = \{D_1, \dots, D_n\}$  be the inspected corpus. Then the *term coverage*  $tcov$  is defined as

$$tcov = \frac{|T'|}{|\cup_{i=1, \dots, n} to(D_i)|}.$$

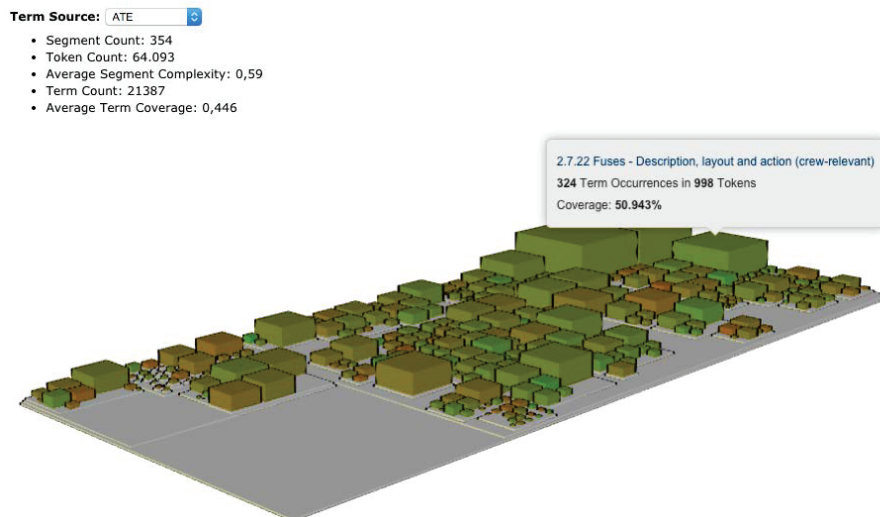


FIGURE 2.5 | Term city showing a term source with high coverage.

A high term coverage implies that the term source fits the document well. The metrics can be visualized using the city metaphor (See Figures 2.5 and 2.6), where a document corresponds to a city and the hierarchical chapter structure is represented

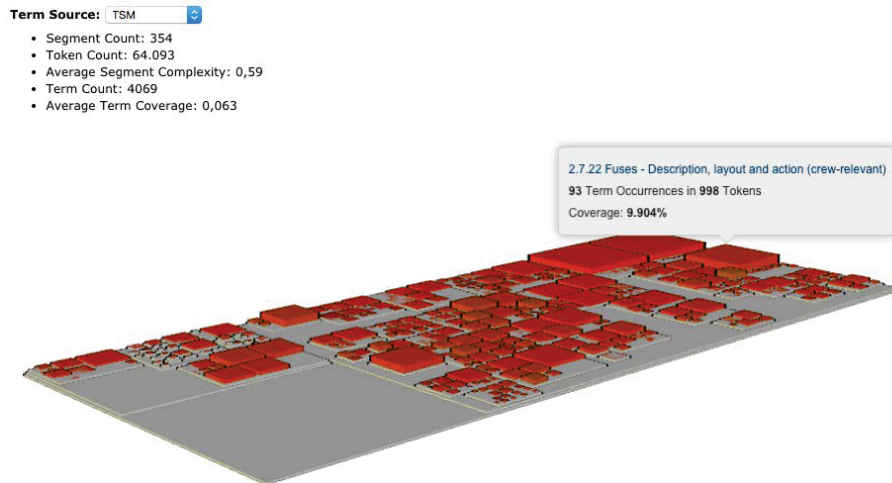


FIGURE 2.6 | Term city showing a term source with low coverage.

by districts and houses. The houses' areas are adapted in accordance to the number of tokens in a segment, while their height and color indicate the respective term coverage. Tall and green houses represent a good term coverage (see Figure 2.5), red ones a low coverage respectively (see Figure 2.6).

### Document Classification and Subject Analysis

In the scope of Linked Data Integration or more specifically the Data Translation of written (technical) texts to an RDF representation, Document Classification and Subject Analysis (Subject Indexing) are additional important Text Analytics methodologies. Unlike the Entity Recognition approaches presented in the previous section these methodologies aim on translating a given text to one or few representative(s). In the context of Linked Data Integration these representatives are usually concepts from an existing Linked (Enterprise) Data graph and are intended to represent the main subject (topic) of the respective document. Different approaches exist for Document Classification and Subject Analysis, ranging from rather knowledge-based approaches to methods based on Machine Learning. This thesis describes a semantification approach for technical documents that ultimately aims on connecting reasonable parts of a technical document to elements of a Linked Enterprise Data graph. Therefore, the Document Classification and Subject Analysis problems are not detailed at this point. The reader might refer directly to Section 4.6 and Section 4.7.

## 2.4 Semantic Information Retrieval

In recent years, semantic technologies entered the market of enterprise applications. In the context of Linked Enterprise Data, Semantic Information Retrieval is considered to be one of the most valuable applications [21]. This is due to the fact that Semantic Information Retrieval can support tasks in all business processes that are regularly concerned with finding important information. Examples for respective tasks can be found in many divisions ranging from Research and Development to Marketing and After Sales. Hence, advanced Information Retrieval methods are emerging for searching relevant resources. This section gives an brief overview of Semantic Information Retrieval.

A plethora of architectures for Semantic Search Engines have been proposed [80, 5, 43, 117, 191]. Mangold [125] surveyed the topic and proposed a classification scheme for Semantic Search approaches. According to Mangold, Semantic Search Engines can be differentiated on the basis of the following characteristics:

- **Architecture:**

A basic differentiation can be made by considering the architecture type of a Semantic Search Engine. In general, two main types exist: Stand-alone applications and meta search engines. While the latter are invoking other search engines, stand-alone applications are actually operating on semantic information. [125]

- **Transparency:**

Another important aspect of Semantic Search Engines is their degree of transparency. Mangold differentiates different transparency grades that express the amount of user interaction that is required to clarify a query. Fully transparent engines appear like ordinary textual search engines and do not require any user-based clarification. Interactive systems, instead, require user actions in order to clarify his input. This process is also referred to as interactive alignment [72] and realized for instance in Semantic Autocompletion components [95]. [125]

- **User Context:**

The user plays a vital role in all information retrieval systems. Mangold proposes to differentiate Semantic Search Engines according to their handling of user context information, e.g. a user's search history, the current location of the user, and the device he is using. Learning systems dynamically adapt their search results on basis of the users context. Hard-coded systems, instead, just follow strict rules. [125]

- **Query Modification:**

The main benefit of Semantic Search Engines is that they can exploit ontological information in order to improve a user's query. The goal of this modification is to consider additional relevant aspects based on the information manifested in the respective ontology. Mangold differentiates three types of query modification. The most basic type requires the user to modify a query manually. More advance approaches rewrite the query by augmenting, trimming and substituting the concept set originally defined by the user. Another type of query modification directly operates on the Linked Data graph (ontology) by traversing with a spreading-activation algorithm. [125]

- **Ontology Technology:**

Mangold also proposes to differentiate systems according to the types of ontologies they support. He claims that most systems operate on F-Logic, RDF or OWL ontologies. [125]

- **Ontology Structure:**

The performance of a Semantic Search Engine often depends heavily on the underlying ontology. Hence, Mangold proposes to differentiate systems according to the ontology structure they are operating on. Systems that operate on anonymous structures are uninformed following all edges in a Linked Data graph. In contrast, engines that work on properties of standard ontologies like SKOS [139] can follow edges in a more targeted manner. The best search results are usually generated by systems that consider domain-specific properties. However, these systems are then especially tailored for the respective domain and can not easily be applied to other problem domains.

- **Document Coupling:**

For this thesis, the most important characteristic of Semantic Search Engines is their type of coupling to the document corpus. Mangold claims that systems with loose and tight coupling exist. Tightly coupled systems require documents to provide metadata that describe their target nodes in a Linked Data graph (ontology). Loosely coupled systems, instead, just apply Entity Recognition techniques and must be considered to be a semantic “augmentation” of traditional, textual search engines.

In the context of this work, stand-alone Semantic Search Engines that work on domain-specific RDF or OWL ontologies are considered. The respective systems must have a tight coupling with the document corpus and support ontology-based query modification. The degree of transparency and the consideration of a user’s context does not directly affect this work.

Figure 2.7 shows the architecture of a Semantic Search Engine. According to this architecture a user interacts with a *user interface* component that essentially provides a search slot and lists search results. The textual inputs of the user are forwarded to a *clarification* component. This component might realize Semantic Autocompletion as proposed by Hyvonen [95] in order to suggest the user concepts from an ontology for defining his query. The suggestions are generated on the basis of the textual input of the user, concepts that have already been included in the query and ontological information from the Linked Data graph. When the query has been fully defined it is first used to update the *user context* in a respective component. Additionally, it is sent to a *retrieval* component. The retrieval component then requests the user context (e.g. the complete search history). The user context and the user defined query are forwarded to a *query modification* component that exploits the provided data in order to generate an augmented version of the query. The augmentation is either based on query rewrite approaches or on graph traversal with spreading-activation algorithms. In either scenario, information from the underlying Linked Data graph (ontology) is considered. When the query has been properly modified, the retrieval component performs a look-up by exploiting the meta information that is used for tightly coupling documents to the Linked Data graph. The resulting resources (documents) are ranked by the retrieval component and finally returned as search result to the user interface for presentation to the user.

*Semantic Assistants* [195] are considered to be a logical extension to Semantic Search Engines. They usually use Speech Recognition methods in order to identify



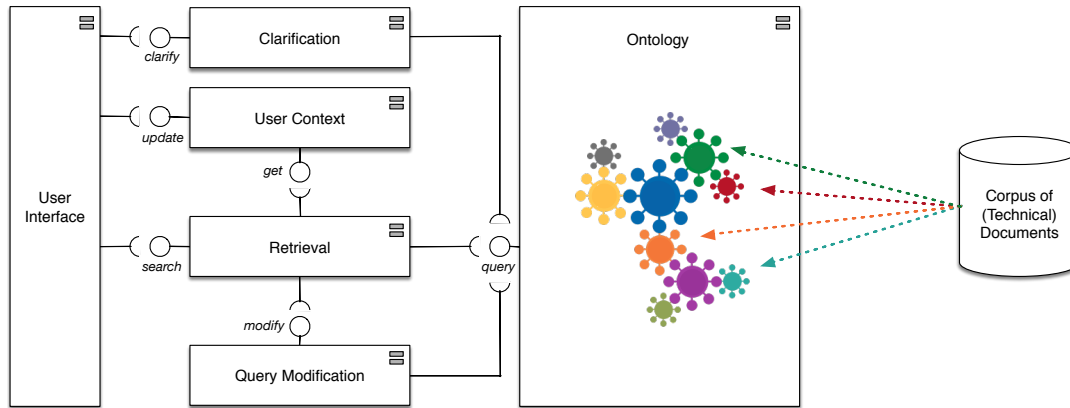


FIGURE 2.7 | Architecture of a Semantic Search Engine.

typical search patterns spoken by the user. When a respective pattern gets matched, a query is generated automatically and handed over to an underlying Semantic Search Engine. Depending on the implementation and the granularity and depth of the semantified resources, Semantic Assistants are able to directly answer a user's question, cf. Question Answering [121, 145].

Semantic Search Engines and Semantic Assistants are powerful information retrieval tools. However, their performance heavily depends on the semantic preparation of the underlying corpora of documents. In order to integrate documents properly into Linked (Enterprise) Data graphs, respective semantic representations are necessary. Chapter 3 describes semantic representations for (technical) documents that facilitate the easy integration into Linked (Enterprise) Data graphs. As large corpora of legacy documents exist that can not be easily transformed to a semantic representation, Chapter 4 presents a holistic semantification approach for technical documents. Chapter 5 gives implementation remarks for realizing the semantification process and Chapter 7 reports on actual semantification projects.



## Chapter 3

# Deep Semantics for Technical Documents

### 3.1 Overview

Today, many proprietary and open formats exist for authoring technical documents. While in former times the single application scenario of technical documentation was the paper-based information of service technicians, new usages arise with technological developments like semantic search applications or smart semantic assistants. These added application scenarios require technical documents to be *accessible* and *linkable*. However, a standardized assessment of the accessibility and linkability of technical documents is hardly possible. The plethora of different formats and undefined assessment criteria are present obstacles. Section 3.2 introduces a 5-STAR maturity schema for assessing the accessibility and linkability of technical documents. The schema clearly states requirements and describes the added values through better accessibility and linkability for each maturity level.

Decent accessibility and linkability usually goes hand in hand with the usage of structured information models. However, a commonly accepted standard for the definition of technical documentation does not yet exist. Instead, a lot of different formats and specifications exist. Their usage is often subject to industry or geographical location. The benefits and downsides of certain formats or specifications are often not clear to decision-makers. Section 3.4 presents the most important information models for the definition of technical documents and discusses their accessibility and linkability with respect to the 5-STAR maturity schema.

Existing information models have limitations regarding accessibility and linkability. Thus, Section 3.3 investigates the absence of an abstracting (meta) ontology for information models in the field of technical documentation. Additionally, TEKNO (TEchnical KNowledge Ontology) is proposed as such an abstracting meta ontology. It is also shown that using TEKNO as an abstraction layer provides additional possibilities for identifying strong technical knowledge that has been inaccessible before.

### 3.2 5-STAR Technical Documentation<sup>1</sup>

Technical documentation has been playing a vital role from the very beginning of mechanical engineering. Although Archimedes and Aristoteles were amongst the first who published technical writings, the machine books of Renaissance era engineers are considered to be the early beginnings of serious technical documentation. Leonardo da Vinci's historical notes and drawings of his machines are popular examples that were supposed to support the reproduction and operation of different equipments. These documents contain structures and elements like exploded view drawings that are still in use in modern documentation.

Since then, technical documentation evolved enormously and was guided by other important progresses like the invention of typewriters or personal computers. In the early 1990s, with the beginning of computer aided authoring of technical documents, the era of electronic documents has started. Nowadays, complex content management systems are employed to meet the ever increasing demands on technical documentation. These requirements arise, for example, due to changes in legal frameworks but also due to the increasing complexity of the respective machinery. While in early years a single document was able to fulfill information needs, today

---

<sup>1</sup>This section and its subsections are significantly extended and revised versions of the contents of the following published article: Sebastian Furth, and Joachim Baumeister. "On the Semantification of 5-Star Technical Documentation." Proceedings of the LWA 2015 Workshops: KDML, FGWM, IR, and FGDB, 2015 [64].

separate documents are necessary for different purposes and target groups. The increasing requirements on technical documentation pose new challenges to authoring processes. Therefore, content reuse through modularization, metadata annotations and translation-oriented writing are established methods for the modern authoring of technical documentation.

However, technical documents written electronically in the past decades must usually be categorized in different epochs. According to empirical observations, technical documents can usually be roughly clustered to three main epochs:

- **Epoch 1:** Single books
- **Epoch 2:** Books assembled from modules
- **Epoch 3:** Annotated modules (Dynamic Documentation)

With respect to accessibility, technical documents from these epochs have substantial differences. While dynamic documentation realized as annotated modules provide high accessibility for service technicians and machines it decreases for unannotated modules and single books. Besides lacking semantic annotations and modularization documents from epochs one and two have other shortcomings, for instance different—partially proprietary—legacy formats or inconsistent formatting guidelines.

It is usually hard to determine the accessibility for a complete corpus. Therefore, this thesis proposes a maturity schema for assessing the accessibility of technical documentation data. The schema lists a number of quality criteria building on each other. For each criterion one star is given; that way, the maturity of documentation data can range from one star to five stars. The schema is inspired by the idea of evaluating the quality of data in the linked open data cloud [16, 100], and was adapted to the needs of technical documentation. The aims of the schemes, however, are identical: First, users should obtain an intuitive impression about the maturity/accessibility of their data; second, users should get motivated to increase the number of stars of their data by adding more semantics.

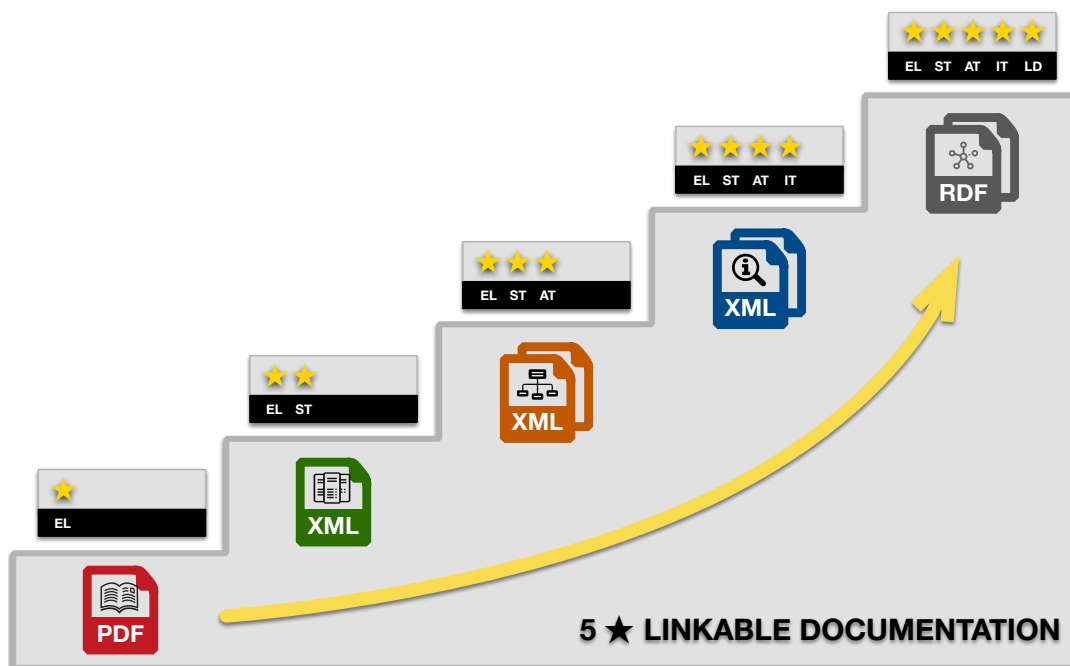


FIGURE 3.1 | The levels of the 5-STARs maturity schema for technical documentation.

The schema for *5-STAR Technical Documentation* data is depicted in Figure 3.1. The first star is given, when the documentation is accessible in an electronic format, for instance, as PDF or Microsoft Word. The documentation gets two stars, when it is accessible in a structured and non-proprietary format, e.g. XML or SGML. Three stars are received for documentation that is available in atomic modules and is free from duplicates. Documentation with four stars provides URIs and information types for these atomic modules. Five star documentation adds semantics to the relevant elements by attaching metadata to the elements that refers to concepts of an ontology. That way, a book itself, particular chapters, and paragraphs can be clearly named and thus can be linked by external applications. Using an ontology enables the automated interlinkage of document elements by using the same concepts of the ontology. Also external ontologies with similar semantics can be aligned to the used ontology. The following sections describe the different maturity levels in detail and discuss benefits and costs for realizing them.

### 3.2.1 1-STAR: Electronic Format

The first level of the maturity scheme requires technical documents to be provided in an electronic format that is *somehow* electronically accessible or can be transformed into such a format easily and reliably. However, the electronic format must at least allow access to the following basic document structures/elements:

- **Pages:** a single page of a document.
- **Blocks:** contiguous text within a single page that can usually be visually separated from other text elements, e.g. headlines or paragraphs.
- **Texts:** contiguous tokens that form a self-contained element, e.g. headline text or sentences.
- **Tokens:** single words.

Additionally, the electronic format must provide basic formatting and positioning information for the aforementioned elements, e.g. font, font size, width, height, and position. Technical documentation provided in the popular PDF format is a typical example of 1-STAR documentation. Although the content is not always directly accessible, it can easily be transformed into an electronic format that provides access to the respective elements (for more details please refer to Section 4.3).

The benefits of technical documentation provided in an electronic format comprise the ability of mass media reproduction at (almost) no costs or more generally the easy way of publishing documentation. Additionally, 1-STAR technical documents are usually easy to exchange, e.g. due to the availability of standard viewers and the fact that normally no explanation is necessary to enable users to read the documents.

1-STAR technical documents can usually be provided without additional efforts as the corresponding formats are widely supported and can easily be exported by a wide range of authoring applications. However, the content of technical documentation that is provided in a 1-STAR electronic format is locked-up from a semantic perspective. Custom parsers are necessary to extract required information from the corresponding documents. Moreover, the extraction results are usually prone to errors which complicates their subsequent usage. The linkability is also limited. Usually only the complete document or at most single pages from a document can be referenced. However, resources like pages are usually not uniformly identifiable. Table 3.1 summarizes the characteristics of 1-STAR technical documentation.

Maternity Level	1-STAR
<b>Accessible Elements</b>	<ul style="list-style-type: none"> <li>– Pages</li> <li>– Untyped Micro Structures (Blocks, Texts)</li> <li>– Nano Structures (Tokens)</li> </ul>
<b>Benefits</b>	<ul style="list-style-type: none"> <li>– Mass media reproduction</li> <li>– Easy publishing</li> <li>– Easy exchange</li> <li>– No need of explanation</li> </ul>
<b>Accessibility</b>	low (requires custom parsers with uncertain results)
<b>Linkability</b>	low (reference to complete document or pages)

TABLE 3.1 | 1-STAR: Maturity Fact Sheet

### 3.2.2 2-STAR: Structured Content

The second level of the maturity scheme requires technical documents to be provided in a structured format that is easily accessible. In contrast to 1-STAR documentation it grants access to more fine-grained and basically typed content structures like headlines on different levels, ordered and unordered lists, tables and others. Additionally, the underlying format must give detailed information about the formatting of these structures. Popular examples of formats fulfilling the requirements of the second levels are arbitrary XML [26] and SGML [76] documents.

The benefits of technical documentation provided in a structured 2-STAR format comprise the easier processing of information due to structured accessibility. Additionally, 2-STAR data can, due to its structured nature, easily be transformed to other formats for publishing or presentation purposes like PDF (via XSL-FO [154]) or HTML. The publication process for 2-STAR technical documentation usually remains simple.

Unlike 1-STAR technical documents the second level requires some additional effort at authoring time. The aforementioned micro structures (headlines, lists, tables etc.) need to be defined and used. This usually involves the creation of corporate authoring guidelines or style guides and subsequently the usage of a dedicated authoring system. Hence, technical documentation fulfilling the requirements of the second level of the maturity scheme usually exists for newer documents that have been produced using a technical authoring system or specially tailored XML editors. Although the second maturity level brings improvements regarding the accessibility huge parts of the information is still locked-up in the documents. The access to information on a granular level still requires custom parsers that are able to infer hierarchies of macro structures like chapters, sections, or subsections. The linkability is usually significantly improved as structures on all levels can receive unique identifiers. The `id` attribute of standard XML is a prominent example for such an identifier on element level. Table 3.2 summarizes the characteristics of 2-STAR technical documents.

<b>Maturity Level</b>	<b>2-STAR</b>
<b>Accessible Elements</b>	Typed Micro structures like headlines, lists or tables
<b>Benefits</b>	<ul style="list-style-type: none"> <li>– Easier processing / less uncertainty</li> <li>– Easy transformation</li> <li>– Still simple publishing</li> </ul>
<b>Accessibility</b>	medium-low (requires custom parsers)
<b>Linkability</b>	medium

TABLE 3.2 | 2-STAR: Maturity Fact Sheet

### 3.2.3 3-STAR: Atomic Modules

The third level of the maturity scheme additionally requires technical documents to be modularized, i.e., split into separate atomic modules describing a self-contained piece of technical information. Modules must be normalized / deduplicated such that information does not exist redundantly. Usually such modules are accompanied by an index file that sequences them in a meaningful order for subsequent publishing or navigation purposes. This way documents are structurally completely broken down. On the macro level chapters, sections, subsections etc. are available as modules. Micro level structures are available from the fine-grained content inside the modules. Modularized technical documents are usually available in an XML- or SGML-based format.

The main benefits of technical documentation provided in a modularized 3-STAR format can be summarized with synergies through content reuse. Such synergies comprise less maintenance and translation efforts and easier integration of third party documentation. Additionally, the modularization grants fine grained, granular access to relevant information on the macro and micro level.

There are additional efforts that are necessary for the authoring of 3-STAR technical documentation, i.e., the creation of accompanying macro structuring information (e.g. index file). This task usually requires the employment of an enterprise content management system that brings support for managing and handling the modular content (modules). Hence, 3-STAR technical documentation is usually rather new as the required enterprise content management systems have merely been introduced during the last decade. Although the availability of 3-STAR technical documents is accompanied by accessibility improvements it is still hard to filter information for specific needs. For example, it still is hardly possible to filter modules with respect to their intended purpose, e.g., distinguish maintenance/repair from descriptive/operational content. The linkability is slightly improved compared to 2-STAR technical documentation as modules are usually uniformly identifiable and are thus directly referencable.

<b>Maturity Level</b>	<b>3-STAR</b>
<b>Accessible Elements</b>	<ul style="list-style-type: none"> <li>– Macro structures like chapters or sections</li> <li>– Typed micro structures like headlines, lists or tables</li> </ul>
<b>Benefits</b>	<ul style="list-style-type: none"> <li>– Less maintenance effort</li> <li>– Less translation effort</li> <li>– Easier integration of third party contents</li> </ul>
<b>Accessibility</b>	medium
<b>Linkability</b>	medium

TABLE 3.3 | 3-STAR: Fact Sheet



### 3.2.4 4-STAR: Information Typed Modules

The fourth level of the maturity scheme additionally requires technical documents to be identifiable and information typed. An information type describes the type of the information, e.g., it allows to distinguish descriptive/operational content from maintenance/repair texts. Therefore, macro and micro structures need to be marked with corresponding information types that are able to express the rhetorical aspect of the respective piece of text. This is usually realized by adding meta tags to respective elements. Usually the information typing through tagging with metadata does not happen for fine grained micro structures but for macro structures / modules (e.g. chapters or sections). Additionally, information typed macro and micro structures need to be identifiable, i.e. each information typed structure needs to be assigned to an unique and stable identifier.

The main benefits of 4-STAR technical documentation is an improved accessibility with respect to rhetorical filtering, i.e. readers are able to constrain the retrieval of documents to certain information types. The mandatory unique identifier for information typed structures leads to a general improvement of linkability and referability. Hence, users might easily bookmark information for later usage or provenance information can easily be added to information extracted from respective structures. The requirement of a stable identifier guarantees the referability across document versions.

The creation of 4-STAR technical documentation involves additional efforts. First, the information typing requires an information taxonomy. An information taxonomy describes the different types of information and how they can be hierarchically structured, e.g. that assembly and disassembly information are specialized repair information. The availability of 4-STAR technical documentation brings large accessibility improvements as structures on different levels provide information about their type and can additionally be clearly identified and referred to. However, the technical documentation is not yet accessible with respect to information about the machinery. For example, users are not yet able to access all information about specific components or functions of a machine. Table 3.4 summarizes the characteristics of 4-STAR technical documentation.

Maturity Level	4-STAR
Accessible Elements	<ul style="list-style-type: none"> <li>– Information typed macro structures</li> <li>– Information typed micro structures</li> </ul>
Benefits	<ul style="list-style-type: none"> <li>– Rhetorical filtering</li> <li>– Improved referability</li> </ul>
Accessibility	medium-high
Linkability	high

TABLE 3.4 | 4-STAR: Fact Sheet

### 3.2.5 5-STAR: Linked Information Units

The fifth level requires atomic, deduplicated, information typed modules that are annotated with concepts describing relevant aspects of the corresponding machinery. The respective components are usually derived from an ontology that describes the underlying machine. Such ontologies are often referred to as digital twins, i.e. semantic representations of products, systems, and processes [170, 73, 168].

Accessibility benefits are realized by additionally annotating macro and micro structures with relevant concepts from these ontologies. This guarantees state-of-the-art accessibility for human users and machines. Employed in a semantic information

system, problem-oriented, targeted access to document structures becomes possible. Additionally, 5-STAR technical documentation opens exploration possibilities, e.g. discovering similar/related content by exploiting the linked concepts. 5-STAR technical documentation is usually fully integrated in Linked Enterprise Data [93, 4].

The annotation of structures with such concepts usually requires elaborated state-of-the-art information management systems. Besides the authoring of technical content it is additionally necessary to maintain the respective ontologies (digital twins), i.e. ontologies must be updated when the underlying machinery changes. Additionally, links between document structures and concepts must be established and maintained.

<b>Maturity Level</b>	<b>5-STAR</b>
<b>Accessible Elements</b>	<ul style="list-style-type: none"> <li>– Information typed and annotated macro structures</li> <li>– Information typed and annotated micro structures</li> </ul>
<b>Benefits</b>	<ul style="list-style-type: none"> <li>– problem-oriented filtering</li> <li>– targeted access</li> <li>– exploration possibilities</li> </ul>
<b>Accessibility</b>	high
<b>Linkability</b>	high

TABLE 3.5 | 5-STAR: Fact Sheet

### 3.3 The Technical Knowledge Ontology

The 5-STAR maturity scheme described in the previous section shows the gradual increase of semantics in technical documents. Each maturity level adds additional aspects to the semantic descriptions of technical documents. This requires an adequate ontological vocabulary that provides support for the required representations. The main requirements of such an ontology are:

- Support the representation of structural components on different levels (*1-STAR, 2-STAR, 3-STAR*)
- Support information typing (*4-STAR*)
- Support annotating with concepts from an external ontology (*5-STAR*)
- Support the integration of existing information models

Additionally, the ontology needs to be able to infer pieces of text that carry strong technical knowledge. Such an inference must work upon existing structural representations and assigned information types. Therefore, the following sections introduce the Technical Knowledge Ontology (TEKNO). This ontology can basically be partitioned into three parts:

1. **Structural Components:**

Describe structural text components in forms of nano, micro and macro structures.

2. **Information Types:**

Describe different types of information that might exist in technical documents.

3. **Core Documentation Entities:**

Describe information typed structures that carry strong technical knowledge.

Figure 3.2 shows an overview of the base components of the TEKNO ontology: the classes `tekno:NanoStructure`, `tekno:MicroStructure` and `tekno:MacroStructure`, `tekno:InformationType` and `tekno:CoreDocumentationEntity` as well as the properties `tekno:next`, `tekno:previous`, `tekno:broader` and `tekno:narrower`. The depicted

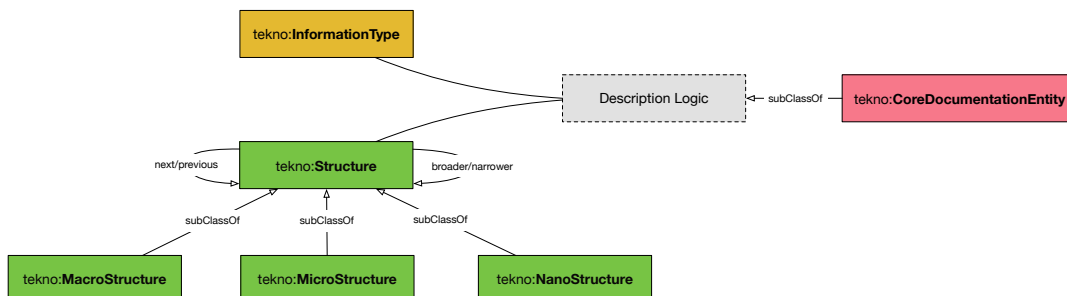


FIGURE 3.2 | Overview of the TEKNO ontology.

classes get subclassed to represent particular nano, micro and macro structures like `tekno:Token`, `tekno:Punctuation` and `tekno:Headline`, `tekno:Paragraph`, `tekno:List` or `tekno:Table` and `tekno:Book`, `tekno:Chapter`, and `tekno:Section` respectively. A

document is semantically represented as a hierarchy of nano, micro, and macro structures assembled of respective instances using `tekno:broader` and `tekno:narrower`. The correct order of nano, micro and macro structures is assured through the properties `tekno:next` and `tekno:previous`. Information types describe the kind of content a structural component carries, e.g. descriptive content or content that enables a reader to perform maintenance or repair tasks. With the availability of information typed micro and macro structure instances, constructs of description logics can be used to formulate *Core Documentation Entities* which represent the aforementioned structures that carry strong technical knowledge.

The remainder of this section is structured as follows: Section 3.3.1 gives an overview of nano and micro structures that typically occur in technical documents, how they are used to assemble macro structures and how macro structures build hierarchies in technical documents. Section 3.3.2 introduces the most important information types that are necessary to represent rhetorical aspects of technical documents. Section 3.3.3 describes *Core Documentation Entities*, i.e. structures that can be inferred from information typed macro and micro structures carrying strong technical knowledge.

### 3.3.1 Structural Representation

Considering only the pure structural composition of a document, the required vocabulary is rather independent of the underlying problem domain. However, the structural decomposition of a document is three-fold. The first part represents token-level elements, i.e. single words or punctuations. In the following section these token-level elements are referred to as *nano structures*. The second part represents different types of block elements that are formed from nano structures, e.g. paragraphs consisting of words. These block-level elements are henceforth called *micro structures*. Finally, macro structures are another form of structural components that are either sequences of micro structures (e.g. sections, see Section 3.3.1) or merely structural elements that are composed from other macro structures (e.g. chapters, see Section 3.3.1). Figure 3.3 shows the different types of structural components. The following sections describe these types in more detail and introduce them formally.

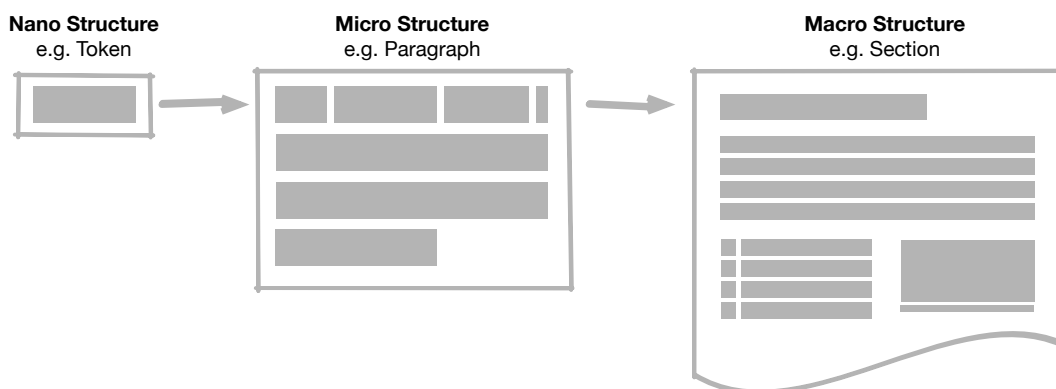


FIGURE 3.3 | Overview of the different types of structural components.

#### Nano Structures

Basically, all texts are assembled from a disjunct set of single words and punctuations. In the following these elements are referred to as *nano structures*. These structures

build the lowest elements in a hierarchical decomposition of a document. Nano structures are assembled to form sentences or other document components on micro level, e.g. lists or tables. A nano structure is formally defined as follows:

**Definition 3.3.1** (Nano Structure). A nano structure is a token-level element  $n$  from the universal set of all nano structures  $N$  such that a document  $D$  from the universal set of all documents (corpus)  $K$  can be decomposed into a disjoint set of nano elements  $D = \cup n$ . A nano structure is represented as a triple  $n = (t, b, e)$  where  $t$  is the type of the nano structure,  $b$  the beginning index of the structure and  $e$  the end index of the structure in the document.

### Micro Structures

Considering the micro structure, a document can be decomposed into a disjunct set of block elements. Micro structures are typically the smallest units in a document that are able to carry a piece of information. However, micro structures are not required to be self-contained, i.e. understandable without contextual information. Typical ex-

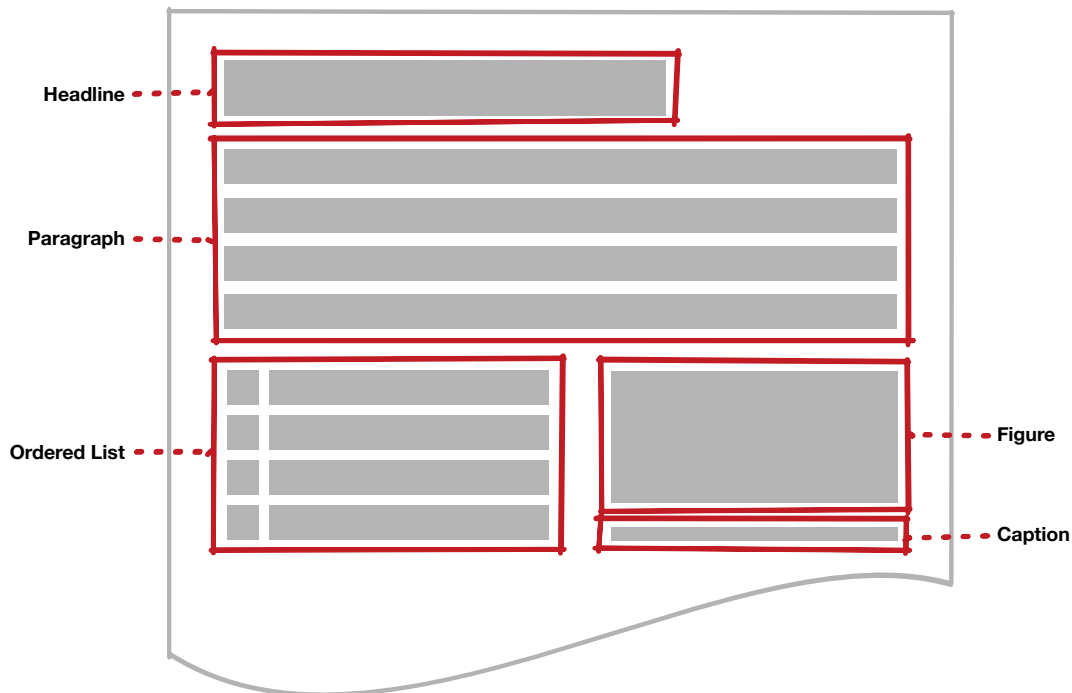


FIGURE 3.4 | Examples of micro structures.

amples for micro structures comprise headlines on different levels, paragraphs, tables, legends, ordered and unordered lists, check lists, figures or captions (see Figure 3.4). A micro structure is formally defined as follows:

**Definition 3.3.2** (Micro Structure). A micro structure is a block-level element  $b$  from the universal set of all micro structures  $M$  such that a document  $D$  from the universal set of all documents (corpus)  $K$  can be decomposed into a disjoint set of block elements  $D = \cup b$ . A micro structure is represented as a triple  $b = (t, b, e)$  where  $t$  is the type of the micro structure,  $b$  the beginning index of the structure and  $e$  the end index of the structure in the document.

Considering an existing document, micro structures usually need to be sequenced, i.e. a reading/processing order must be defined for the sequence of block-level elements. The reconstruction of the originally intended reading order is especially in unstructured formats (e.g. PDF) a challenging task. The micro structure ordering is formally defined as follows:

**Definition 3.3.3** (Micro Structure Ordering). The order of block-level elements  $b$  from  $M = \{b_1, b_2, \dots, b_n\}$  is defined pairwise using the predicates  $next(b_i, b_{i+1})$  and  $previous(b_{i-1}, b_i)$ .

### Macro Structures

Sequences of micro structures form *macro structures*, i.e., self-contained pieces of text that are able to fulfill information needs without additional contextual information. Typical examples of macro structures in classical publications comprise sections, chapters or even books. A macro structure is formally defined as follows:

**Definition 3.3.4** (Macro Structure). A macro structure  $s_{i,k}$  from the universal set of all macro structures  $S$  consists of an ordered number of block-level elements  $b$  from micro structures  $M$  such that  $s_{i,k} = (b_i, b_{i+1}, \dots, b_k)$ . A macro structure  $s$  is self-contained, i.e., a human reader is able to understand the textual content of  $s_{j,k}$  without necessarily reading another macro structure  $s_{l,m}$ .

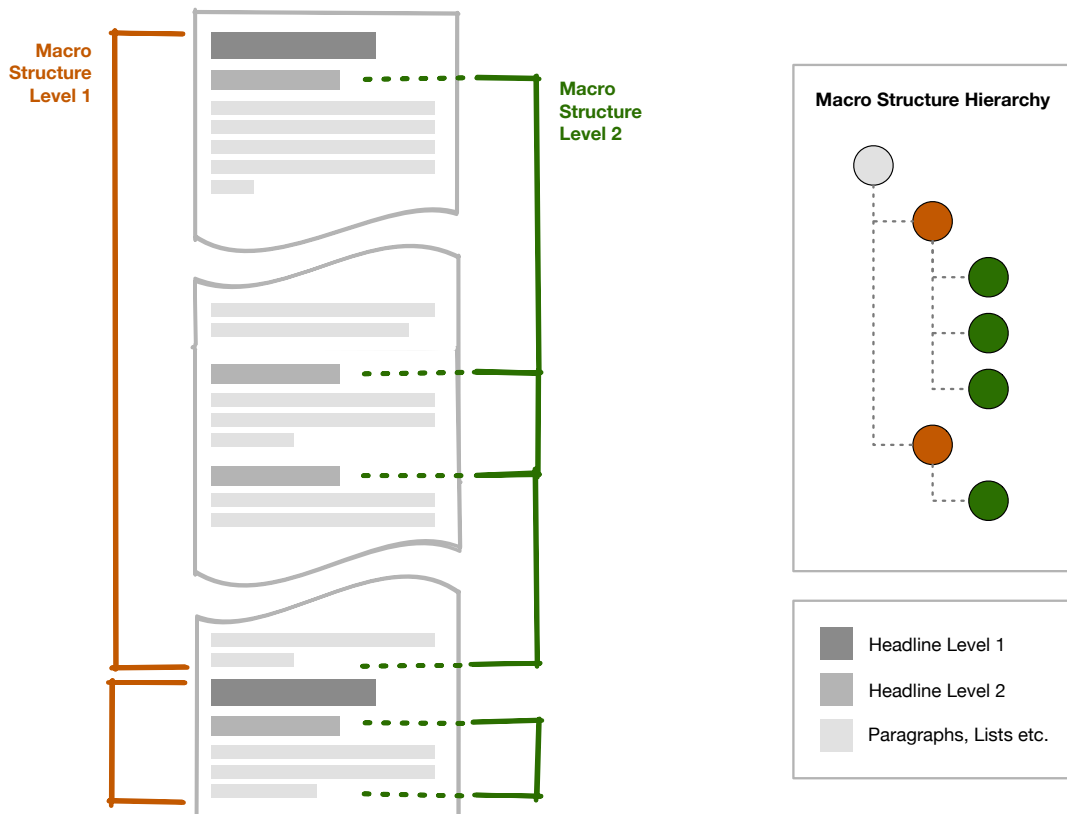


FIGURE 3.5 | Examples of macro structures.

The consumption of single macro structures without tool support is in most cases not reasonable. Therefore, macro structures usually get assembled in sequences or hierarchies for publishing or navigational purposes. Figure 3.5 shows an example of

hierarchically assembled macro structures. Macro structure hierarchies are formally defined as follows:

**Definition 3.3.5** (Macro Structure Hierarchies). Macro structures  $s$  from the universal set of all macro structures  $S$  can form hierarchies. Given a more general macro structure  $s_i \in S$  and a more specific macro structure  $s_j \in S$  the hierarchy is defined pairwise using the predicates *broader*( $s_i, s_j$ ) for the child-to-parent relation and *narrower*( $s_j, s_i$ ) for the parent-to-child relation respectively.

### Related Work

The Document Ontology schema of the SALT ontology [78], and the pattern ontology [53] are popular examples for the structural description of (scientific) publications. Şah and Wade [171] proposed an ontology that covers a reasonable subset of the DocBook standard. The most important elements of this ontology are `docbook:Book`, `docbook:Article`, `docbook:Chapter` on the macro level and block elements (micro level) like `docbook:Paragraph`, `docbook:Procedure`, and `docbook:Figure`.

### 3.3.2 Information Types

In contrast to the structural organization of a document, information types are used to model the rhetorical structure of technical documents. Information types are formally defined as follows:

**Definition 3.3.6** (Information Type). An information type  $t$  from the universal set  $T$  of all information types is defined as a singleton concept instance that can be assigned to macro structures  $i \in I$  and micro structures  $m \in M$  using the predicates *type*( $t, i$ ) and *type*( $t, m$ ) respectively.

Information types can form taxonomies where successor and ancestor nodes represent specialized and more general types respectively. An information type hierarchy is formally defined as follows:

**Definition 3.3.7** (Information Type Hierarchies). Information types  $t_i$  from the universal set of all information types  $T$  can form hierarchies. Given a more general information type  $t_i \in T$  and a more specific information type  $t_j \in T$  the hierarchy is defined pairwise using the predicates *broader*( $t_i, t_j$ ) for the child-to-parent relation and *narrower*( $t_j, t_i$ ) for the parent-to-child relation respectively.

The TEKNO ontology defines the following information types to cover the most common rhetorical aspects of technical documents. The information types are subclassed where needed.

- **General Information:**  
General aspects of the document or the machine in focus.
- **Index:**  
Indices like table of contents, subject catalogs, list of abbreviations etc.
- **Safety Instructions:**  
Safety notes to be obtained while working with the machine.
- **Description:**  
Information about specific components or functions.

- **Operation:**  
Information about the usage of the machine, specific components or functions.
- **Maintenance:**  
Information about maintenance works, schedules etc.
- **Adjustment:**  
Information about necessary adjustments in specific situations.
- **Repair:**  
Repair procedures; important subclasses are **Assembly** and **Disassembly**
- **Test:**  
Information about testing the correction functioning of the machine.
- **Fault Isolation:**  
Detailed troubleshooting information.
- **Parts:**  
Spare part information.

The minimum level on which reasonable information type assertions can be made are micro structures. As macro structures are assembled from micro structures the rhetorical aspects can change multiple times within a single macro structure. An example are safety instructions that occur at the beginning of a module that describes the disassembly of a component. The safety instruction might be represented as a paragraph that is typed as safety information. The disassembly procedure on the other side might be represented as a procedure that is typed as repair information.

### Information Typed Structural Components

Unlike structural components the beginning and the end of rhetorical aspects can often not be clearly defined. Hence, it is usually not feasible to define an additional structure that represents the rhetorical decomposition of the text. Instead, the TEKNO ontology approximates a complete rhetorical decomposition of a document by assigning information types to instances of macro and micro structures. TEKNO exploits the fact, that core information types like safety instructions or maintenance information can often be linked explicitly to particular structures like chapters, sections, paragraphs or procedures and thus provide the required base data. Figure 3.6 shows an example for information typing textual content on micro level.

The example shows that micro structures appearing in a single macro structure might be assigned different information types. Referring to Figure 3.6 a section might start with some safety instructions, then giving step-wise repair instructions which in turn refer to a figure that describes certain subcomponents. On the macro level the information type might be assigned to complete sections, chapters, and books.

### Related Work

For the representation of scientific articles the Rhetorical Ontology schema of the SALT ontology [78] or the Discourse Elements Ontology [46] provide an appropriate vocabulary. Thus, rhetorical aspects like the motivation, background, methods etc. can be modeled as instances of respective classes.



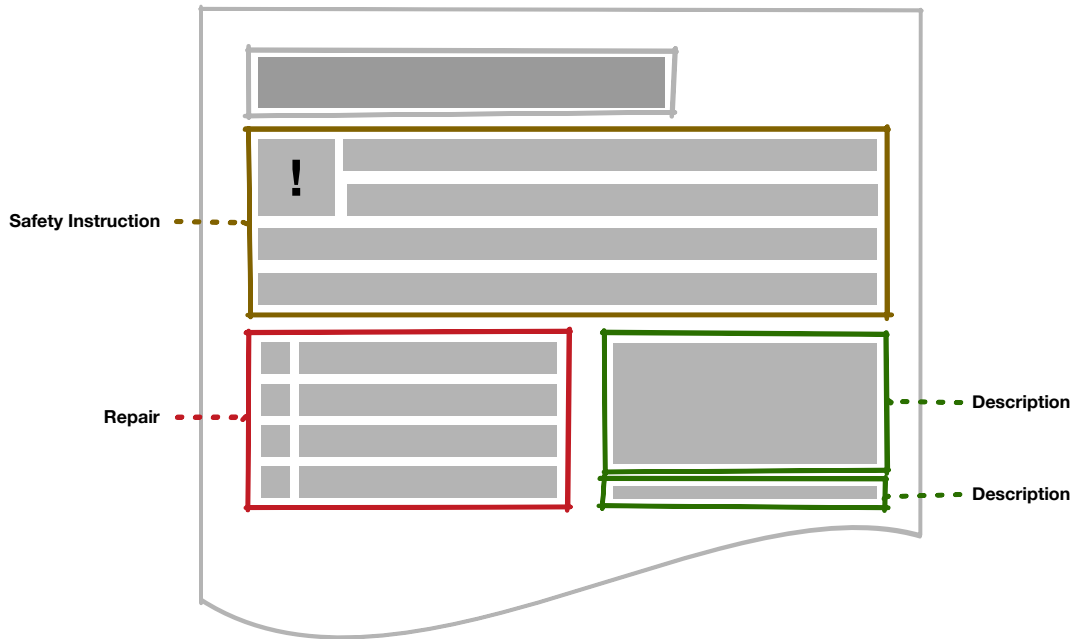


FIGURE 3.6 | Examples of information typed structures.

While the underlying idea also facilitates the rhetorical modeling of technical documentation the concrete classes do not fit the technical domain. For instance law requires technical documentation to follow a certain rhetorical organisation, e.g. safety notes need to precede actual operation instructions. Thus it would be beneficial to semantically represent safety notes.

### 3.3.3 Core Documentation Entities

The main goal of the TEKNO ontology is providing access to pieces of technical documentation that carry strong technical knowledge. Representing the structural and rhetorical aspects of technical documentation is a considerable step in this direction. However, the most important aspects of technical documents are typically interweaved in these two structures. The entropy of such aspects is typically sufficient to satisfy an immediate information need. Therefore, the TEKNO ontology introduces a novel representation for such aspects – *Core Documentation Entities*. Core Documentation Entities combine structural and rhetorical aspects in order to make the respective structures easily accessible. Core Documentation Entities are required to be self-contained, i.e., they must be consumable without additional contextual information. A Core Documentation Entity is formally defined as follows.

**Definition 3.3.8** (Core Documentation Entity). A Core Documentation Entity  $e$  from the universal set of all Core Documentation Entities  $E$  is a self-contained micro structure  $m \in M$  or macro structure  $i \in I$  that is assigned an information type  $t \in T$  that clearly describes its rhetorical aspect.

Typical examples for Core Documentation Entities comprise (dis-)assembly procedures or component overviews. The following sections formally introduce the most important Core Documentation Entities. For better readability the descriptions of the Core Documentation Entities are grouped by their underlying macro and micro structures.

### Paragraph-based Core Documentation Entities

Huge parts of technical documents are filled with paragraphs. However, some of these paragraphs carry specific kinds of information. Direct access to these paragraphs is beneficial. The following sections formally introduce paragraph-based Core Documentation Entities and describe their main purposes.

#### Notes, Warnings, Safety Instructions etc.

Technical documents usually contain elements that point out specific information in order to avoid harm to humans, machines and the environment. Therefore, ordinary paragraphs are often marked by dedicated colors, figures, and special words. Typical examples comprise *notes*, *warnings*, *safety instructions*, *environmental regulations* or *danger* and *attention notes*. Expressions 3.1 - 3.6 show examples of corresponding Core Documentation Entities. The main difference between the following expressions are the different values of the `text` predicate of the nano structures  $a$ . Hence, the following Core Documentation Entities are micro structures of type `tekno:Paragraph` with the information type `tekno:SafetyInstruction` that contain a keyword (nano structure) with respective textual content. Such paragraphs usually occur throughout the complete technical documentation, i.e. in all kinds of technical documents.

$$\begin{aligned} & \{b \in M \mid \text{type}('Paragraph', b) \\ & \quad \wedge \text{type}('SafetyInstruction', b) \\ & \quad \wedge \exists a \in N : (\text{narrower}(a, b) \wedge \text{text}('Note', a)) \} \end{aligned} \quad (3.1)$$

$$\begin{aligned} & \{b \in M \mid \text{type}('Paragraph', b) \\ & \quad \wedge \text{type}('SafetyInstruction', b) \\ & \quad \wedge \exists a \in N : (\text{narrower}(a, b) \wedge \text{text}('Warning', a)) \} \end{aligned} \quad (3.2)$$

$$\begin{aligned} & \{b \in M \mid \text{type}('Paragraph', b) \\ & \quad \wedge \text{type}('SafetyInstruction', b) \\ & \quad \wedge \exists a \in N : (\text{narrower}(a, b) \wedge \text{text}('Safety', a)) \} \end{aligned} \quad (3.3)$$

$$\begin{aligned} & \{b \in M \mid \text{type}('Paragraph', b) \\ & \quad \wedge \text{type}('SafetyInstruction', b) \\ & \quad \wedge \exists a \in N : (\text{narrower}(a, b) \wedge \text{text}('Environment', a)) \} \end{aligned} \quad (3.4)$$

$$\begin{aligned} & \{b \in M \mid \text{type}('Paragraph', b) \\ & \quad \wedge \text{type}('SafetyInstruction', b) \\ & \quad \wedge \exists a \in N : (\text{narrower}(a, b) \wedge \text{text}('Danger', a)) \} \end{aligned} \quad (3.5)$$

$$\begin{aligned} & \{b \in M \mid \text{type}('Paragraph', b) \\ & \quad \wedge \text{type}('SafetyInstruction', b) \\ & \quad \wedge \exists a \in N : (\text{narrower}(a, b) \wedge \text{text}('Attention', a)) \} \end{aligned} \quad (3.6)$$

### Special Tools Description

Special Tool descriptions are an additional kind of paragraph-based Core Documentation Entities. Such descriptions usually occur as `tekno:Paragraph` with the information type `tekno:Parts` or `FaultIsolation` next to a `tekno:Figure` that depicts the respective special tool. Additionally, these special tool descriptions usually occur within part catalogues or trouble shooting documents as these tools are usually required during replacement or diagnosis procedures. Expression 3.7 is a formal definition of a special tool description.

$$\begin{aligned} & \{b \in M \mid \text{type}('Paragraph', b) \\ & \quad \wedge (\text{type}('Parts', b) \vee \text{type}('FaultIsolation', b)) \\ & \quad \wedge \exists a \in M : (\text{next}(b, a) \wedge \text{type}('Figure', a)) \} \end{aligned} \quad (3.7)$$

### List-based Core Documentation Entities

While paragraphs are a rather unstructured kind of micro structure ordered and unordered lists carry much more semantics. The targeted exploitation of such list-based structures grants access to information that can serve as an important source for taxonomy extraction or other information extraction tasks. The following sections formally introduce list-based Core Documentation Entities and describe their main purposes.

#### Component Overviews

Component overviews are used to give a quick and precise description of a certain component. They usually occur in description documents like user manuals. Sometimes they are also depicted in repair manuals in forms of extended technical descriptions. Component overviews are usually realized as exploded view drawings or annotated photos followed by an ordered list (legend) describing the component depicted in the drawing or photo. Thus, the Core Documentation Entity *Component Overview* is defined as micro structure of type `tekno:OrderedList` with the information type `tekno:Description` that immediately follows a micro structure of type `tekno:Figure`. Expression 7.1 is a formal definition of a component overview.

$$\begin{aligned} & \{b \in M \mid \text{type}('OrderedList', b) \\ & \quad \wedge \text{type}('Description', b) \\ & \quad \wedge \exists a \in M : (\text{next}(a, b) \wedge \text{type}('Figure', a)) \} \end{aligned} \quad (3.8)$$

#### Repair / Maintenance / Operating Instructions

Different kinds of instructions are another category of list-based Core Documentation Entities. Such instructions occur in operating manuals, repair manuals, and maintenance plans as operating, repair or maintenance instructions respectively. They give the reader detailed instructions for fulfilling a specific task. List items (nano structures) usually represent single steps the reader has to perform. Instructions

are usually realized as `tekno:OrderedList` or `tekno:UnorderedList` that are information typed with `tekno:Operation`, `tekno:Repair` or `tekno:Maintenance` respectively. Expressions 3.9 - 3.11 are formal definitions of the aforementioned instructions.

$$\{b \in M \mid ( \text{type}('OrderedList', b) \vee \text{type}('UnorderedList', b) ) \wedge \text{type}('Operation', b) \} \quad (3.9)$$

$$\{b \in M \mid ( \text{type}('OrderedList', b) \vee \text{type}('UnorderedList', b) ) \wedge \text{type}('Repair', b) \} \quad (3.10)$$

$$\{b \in M \mid ( \text{type}('OrderedList', b) \vee \text{type}('UnorderedList', b) ) \wedge \text{type}('Maintenance', b) \} \quad (3.11)$$

### Table-based Core Documentation Entities

Tables are another kind of micro structure that represent the encapsulated information in a structured way. Tables in technical documents usually contain rows (header) or columns that describe the information in the table cells. Thus, the targeted exploitation of tables gives access to detailed and precise technical information. The following sections formally introduce table-based Core Documentation Entities and describe their main purposes.

#### Technical Data

Technical Data tables normally describe detailed information about machines, vehicles or specific components. Such tables contain information about the weight, dimensions, speed or other performance indicators. Tables carrying technical data usually occur in descriptive technical documents, e.g., the user manual. Hence, the Core Documentation Entity *technical data* is a `tekno:Table` that has the assigned information type `tekno:Description` and contains nano structures (tokens in table cells) that have special types or texts, e.g., type `tekno:Unit` and text “kg” for weight data. Expression 3.12 is a formal definition of the Core Documentation Entity *technical data*.

$$\{b \in M \mid \text{type}('Table', b) \wedge \text{type}('Description', b) \wedge \exists a \in N : (\text{narrower}(a, b) \wedge \text{type}('Unit', a) \wedge \text{text}('kg', a)) \} \quad (3.12)$$

#### Fuels / Lubricants Table

Two other kinds of tables that often occur in technical documents are fuels and lubricants tables. These tables give detailed information about the amount and types of fuels needed or components requiring special lubricants. The main difference between these two table types are their occurrence in technical documents. While the Core Documentation Entity `tekno:FuelsTable` usually occurs in operating manuals instances of `tekno:LubricantsTable` can usually be found in maintenance documents. Both Core Documentation Entities are usually realized as `tekno:Table` with respective

information types (`tekno:Operation` or `tekno:Maintenance`). Additionally, these tables contain nano structures (tokens in table cells) that can be identified as `tekno:Unit` with specific textual content like “ml” or “l”. Expressions 3.13 and 3.14 are formal definitions of these two Core Documentation Entities.

$$\begin{aligned} & \{b \in M \mid \text{type}('Table', b) \\ & \quad \wedge \text{type}('Operation', b) \\ & \quad \wedge \exists a \in N : (\text{narrower}(a, b) \wedge \text{type}('Unit', a) \wedge \text{text}('l', a)) \} \end{aligned} \quad (3.13)$$

$$\begin{aligned} & \{b \in M \mid \text{type}('Table', b) \\ & \quad \wedge \text{type}('Maintenance', b) \\ & \quad \wedge \exists a \in N : (\text{narrower}(a, b) \wedge \text{type}('Unit', a) \wedge \text{text}('ml', a)) \} \end{aligned} \quad (3.14)$$

### Maintenance Table

Apart from lubrication information, service technicians usually require additional maintenance information. Such information is normally organized in respective tables that list the affected components and maintenance actions that are required in certain periods (daily to yearly). Maintenance tables are usually found in service or maintenance manuals. Hence, the Core Documentation Entity *Maintenance Table* is typically a `tekno:Table` that has the information type `tekno:Maintenance` assigned. Additionally, the Core Documentation Entity requires that certain nano structures occur. Expression 3.15 shows a formal definition of a maintenance table.

$$\begin{aligned} & \{b \in M \mid \text{type}('Table', b) \\ & \quad \wedge \text{type}('Maintenance', b) \\ & \quad \wedge \exists c \in N : (\text{narrower}(c, b) \wedge \text{text}('Item', c)) \\ & \quad \wedge \exists d \in N : (\text{narrower}(d, b) \wedge \text{text}('Daily', d)) \wedge \text{next}(c, d) \\ & \quad \wedge \exists e \in N : (\text{narrower}(e, b) \wedge \text{text}('Weekly', e)) \wedge \text{next}(d, e) \\ & \quad \wedge \exists f \in N : (\text{narrower}(f, b) \wedge \text{text}('Daily', f)) \wedge \text{next}(e, f) \\ & \quad \wedge \exists g \in N : (\text{narrower}(g, b) \wedge \text{text}('Monthly', g)) \wedge \text{next}(f, g) \\ & \quad \wedge \exists h \in N : (\text{narrower}(h, b) \wedge \text{text}('6 - Month', h)) \wedge \text{next}(g, h) \\ & \quad \wedge \exists i \in N : (\text{narrower}(i, b) \wedge \text{text}('Yearly', i)) \wedge \text{next}(h, i) \} \end{aligned} \quad (3.15)$$

### Troubleshooting Table

Troubleshooting tables typically list problems, their potential causes and actions to solve them. They usually occur in special troubleshooting chapters or particular troubleshooting documents. Hence, the Core Documentation Entity *Troubleshooting Table* is a `tekno:Table` having the information type `tekno:FaultIsolation` assigned and contains tokens (nano structures) with specific textual content (e.g. “Problem”, “Cause” and “Action”). Expression 3.16 is a formal definition of a troubleshooting

table.

$$\begin{aligned}
& \{b \in M \mid \text{type}('Table', b) \\
& \quad \wedge \text{type}('FaultIsolation', b) \\
& \quad \wedge \exists c \in N : (\text{narrower}(c, b) \wedge \text{text}('Problem', c)) \\
& \quad \wedge \exists d \in N : (\text{narrower}(d, b) \wedge \text{text}('Cause', d)) \wedge \text{next}(c, d) \\
& \quad \wedge \exists e \in N : (\text{narrower}(e, b) \wedge \text{text}('Action', e)) \wedge \text{next}(d, e)\}
\end{aligned} \tag{3.16}$$

### Measurement Table

A measurement table usually lists electric, pneumatic, and hydraulic components and expected measurement values. Usually, such tables appear in troubleshooting documents, as measurements are a suitable way to identify erroneous items. Often measurement tables are accompanied by schematics visualizing the connections between listed components. Hence, the Core Documentation Entity *Measurement Table* is a **tekno:Table** having the information type **tekno:FaultIsolation** assigned, preceded by a figure and contains tokens (nano structures) with specific textual content (e.g. units like “bar” or “mA”). Expression 3.17 is a formal definition of a measurement table.

$$\begin{aligned}
& \{b \in M \mid \text{type}('Table', b) \\
& \quad \wedge \text{type}('FaultIsolation', b) \\
& \quad \wedge \exists a \in N : (\text{narrower}(a, b) \wedge \text{type}('Unit', a) \wedge (\text{text}('mA', a) \vee \text{text}('bar', a))) \\
& \quad \wedge \exists c \in M : (\text{next}(c, b) \wedge \text{type}('Figure', c)) \}
\end{aligned} \tag{3.17}$$

### Part List

Part lists usually occur in dedicated documents, i.e. spare part catalogues. These documents are often highly structured and follow a simple schema. A drawing of a component is usually accompanied by a table listing the respective parts. Thus, the Core Documentation Entity *Part List* is a table having the information type **tekno:Parts** assigned, preceded by a figure and contains tokens (nano structures) indicating the spare part list, e.g. “Part Number”, “Part Name” or “Quantity”. Expression 3.18 is a formal definition of a part list.

$$\begin{aligned}
& \{b \in M \mid \text{type}('Table', b) \\
& \quad \wedge \text{type}('Parts', b) \\
& \quad \wedge \exists c \in N : (\text{narrower}(c, b) \wedge \text{text}('Part Number', c)) \\
& \quad \wedge \exists d \in N : (\text{narrower}(d, b) \wedge \text{text}('Part Name', d)) \wedge \text{next}(c, d) \\
& \quad \wedge \exists e \in N : (\text{narrower}(e, b) \wedge \text{text}('Quantity', e)) \wedge \text{next}(d, e) \\
& \quad \wedge \exists c \in M : (\text{next}(f, b) \wedge \text{type}('Figure', f)) \}
\end{aligned} \tag{3.18}$$

## 3.4 Information Models for Technical Documents

The previous section introduced the TEKNO ontology, i.e., a semantic vocabulary that is able to structurally and rhetorically break down technical documents. Although this vocabulary is able to semantically represent technical documents and additionally gives access to Core Documentation Entities, it is barely possible to find documents that are created using this vocabulary. TEKNO usually acts as an abstracting meta ontology over structural and rhetorical instances from other information models. Therefore, the following sections give an overview of existing information models that support the authoring of technical documents. Additionally, for each information model a mapping to the TEKNO ontology is presented. This underlines the intention of TEKNO to serve as an abstracting meta ontology and shows that the described inference of Core Documentation Entities can also be realized for technical documents that are written according to existing information models.

### 3.4.1 PI-Mod

PI-Mod [202, 54] is a free and open standard that has been established by a cooperation of research and industry in 2008. The standard provides a semantic information structure for technical documentation. Its target is the standardization of information modeling in mechanical engineering and construction. The core of the standard is the PI classification where pieces of information are classified according to product and information classes. Additionally, the PI-Mod standard focuses on the modularization of technical documentation and supports the integration of third party documentation (e.g. OEM documentation). Another target of the standard is the easy integration in enterprise content management systems.

#### Information Model

The PI-Mod information model is provided as an XML Doctype Definition (DTD) that is accompanied by an element reference and XSL-FO stylesheets for the automatic generation of PDF publications. The information model supports for the modular creation of information and the reuse of content during the publication process.

#### Element Overview

The core elements of the PI-Mod information are *modules* with respective module types. For a more fine-grained creation of semantic content, dedicated XML elements exist. The usage of some of these XML elements is constrained. All modules are subsequently classified using the PI-class mechanism, i.e. with at least an information and a product class. Figure 3.7 depicts an overview of the PI-Mod information model. The following sections describe the respective elements in more detail.

#### Structural Elements

The basic units of information in PI-Mod are modules. The PI-Mod information model contains elements that facilitate the construction of structural representations in such modules. On a macro level, the structuring of modules is implementation dependent and therefore out of scope of the information model. On a micro level, dedicated XML elements exist. Some of these dedicated XML elements are constrained

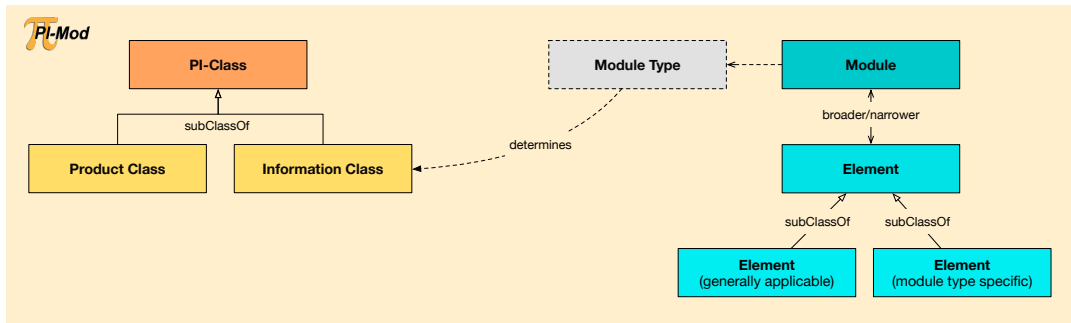


FIGURE 3.7 | Overview of the PI-Mod information model.

to different module types while others are more generally applicable. For example, amongst others the following XML elements can be used in different module types:

- **paragraph:**  
Element for the definition of standard paragraphs.
- **table:**  
Element for the definition of information in tables.
- **note:**  
Element for the definition of notes that shall be obeyed.
- **lists:**  
Element for the definition of lists.
- **safetyadvice:**  
Element for the definition of safety instructions.

### Rhetorical Elements

The PI-Mod specification heavily relies on the PI-Class classification scheme. A central element of this scheme is the classification according to information classes. These information classes then can be utilized to introduce rhetorical aspects to the modules of the technical documentation. A central element of the PI-Class scheme is that it allows for the definition of multiple, hierarchical levels of information classes, i.e. information classes are handled like taxonomies. A taxonomy for information classes might look like this in PI-Class:

- **Maintenance**
  - Safety Instructions
  - Maintenance Plan
- **Operation**
  - Decommissioning
  - Commissioning
- **Description**
  - Explanation
  - Notes



- Intended Use
- **Trouble Shooting**
  - Fault Description
  - Safety Instructions
- ...

Technically, the information class can be added to modules using dedicated XML attributes. The XML attributes can be added to a variety of PI-Mod XML elements.

### Core Documentation Entities

While some of the aforementioned structural XML elements can appear in different module types, the usage of others is limited to certain module types. The following module types exist in PI-Mod:

- **descriptive:**  
Module type for descriptive content
- **task:**  
Module type that is used to express specific tasks
- **task-intervals:**  
Module type that clearly defines intervals of specific tasks
- **tools:**  
Module type that is used for the description of required tools
- **lubrication:**  
Module type for lubrication tasks
- **diagnosis:**  
Module type for diagnosis and trouble shooting task
- **glossary:**  
Module type that is used for glossaries or indices

The module type usually predetermines the information class, e.g. **task** and **task\_body** are often used in conjunction with the information class **Maintenance**. Thus, some of the dedicated XML elements comply with the character of Core Documentation Entities. In the following, this is exemplarily shown for the **task\_body** element that appears in modules with the type **task**. For a complete list of such elements we refer to the PI-Mod specification. Elements that are exclusively applicable to the **task\_body** element are:

- **action:**  
Element for the definition of a specific step/action.
- **or-action:**  
Element for the definition of an alternative step/action.
- **result:**  
Element for the definition of the result of preceding actions.
- **troubleshooting:**  
Element for the definition of trouble shooting information.

### Additional Metadata

The PI-Class classification scheme is a vital part of the PI-Mod specification. It also aims on classifying modules with respect to product classes. Therefore, the PI-Class scheme allows for the hierarchical definition of product classes (taxonomies). These product classes might represent machines and their respective components or other product related domain elements. The following hierarchy is an excerpt of a typical PI-Class product taxonomy.

- **Drive**
  - Transmission
    - \* Gear
    - \* Gear Shaft
- **Hydraulics**
  - Oil Pump
- ...

Analogous to the information classes, these additional metadata elements are added to modules using dedicated XML attributes. These XML attributes are applicable to a variety of PI-Mod XML elements.

### TEKNO Extension: PI-Mod

In general the PI-Mod information model provides similar classes to the TEKNO ontology. From a structural perspective a module assembled from XML elements according to the PI-Mod information model corresponds to a `tekno:MacroStructure` instance with `tekno:MicroStructure` instances respectively. PI-Mod's information class is equivalent to the `tekno:InformationType`. Core Documentation Entities can either be declared directly using XML elements constrained to certain module types or can be derived from PI-Mod elements with assigned information classes and the respective expressions in the TEKNO ontology. The latter requires the aforementioned mapping of the PI-Mod information model to the TEKNO ontology. Figure 3.8 shows the mapping between the PI-Mod information model and the TEKNO ontology. Bold arrows between the two models indicate the mapping.

### Formats and Delivery

Information modeled according to the PI-Mod specification is typically authored in enterprise content management systems. As described before the PI-Mod specification standardizes the micro structuring of modules. Thus, enterprise content management systems usually use their own persistence and macro structuring formats.

### Tool Support

The PI-Mod specification is implemented in a variety of proprietary enterprise content management systems, especially in the German market, amongst others TIM RS (Fischer Computer Technik AG), COSIMA (Docufy GmbH) or SCHEMA ST4 (SCHEMA GmbH). Additionally, the specification comprises XSL-FO stylesheets which can be used to transform PI-Mod XML to PDF.

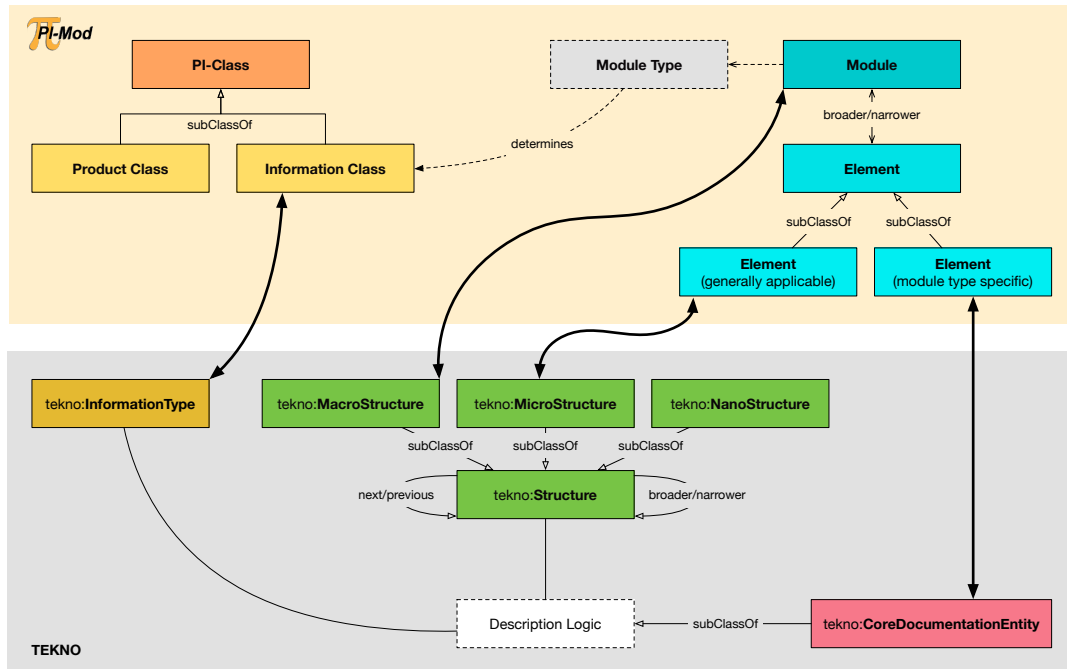


FIGURE 3.8 | Mapping of the PI-Mod information model to the TEKNO ontology.

## Discussion

In general, technical documentation authored according to the PI-Mod information model fulfills huge parts of the 5-STAR maturity schema. Micro structures as required for the first two maturity levels are available through XML elements. The third maturity level requires modularization, which is the fundamental idea of the PI-Mod information model. Independent from actual implementations the only macro structure available is the module. Depending on the implementation other macro structures might be available which enables the definition of macro structure hierarchies. The fourth star requires information typing and identifiability for modules. While information typing is available through the information class of the PI-Class classification, the identifiability of modules is questionable and depends in most cases on the actual implementation. The requirements of the fifth star are also met through the product class of the PI-Class classification scheme which enables the annotation of modules (macro structures) or elements (micro structures) with elements of a product/component taxonomy. In summary, the PI-Mod information model enables the creation of 5-STAR technical documentation. The conjunction of the PI-Mod information model and the Core Documentation Entity expressions of the TEKNO ontology facilitates good accessibility. Lacking specifications for the assembling of macro structure hierarchies might complicate the accessibility of printed documents and the logic navigation through a document. Table 3.6 summarizes this discussion according to the a 5-STAR rating schema.





1-STAR	2-STAR	3-STAR	4-STAR	5-STAR
				

TABLE 3.6 | PI-Mod 5-STAR rating.

### 3.4.2 iiRDS

The iiRDS specification [187] is developed by the Information 4.0 working group of the German Association for Technical Documentation. Members of the working group are from science, industry as well as software and consulting companies. The goal of the specification is to provide a standard that supports the transition from document-based documentation to dynamic documentation. In contrast to traditional document-based publications that bundle all contents in one document, dynamic documentation assembles required information according to available metadata from fine grained information units. Therefore, in iiRDS documentation is a bulk of topics that are represented as metadata. Thus, the aim of iiRDS is the standardization of metadata for technical documentation. A corresponding ontology provides a standardized vocabulary for the creation of contents for dynamic technical documentation. Additionally, the iiRDS standard specifies the structure of delivery packages.

#### Information Model

The iiRDS information model is provided as RDFS [30] ontology and makes use of other (de facto) standards like the DCMI Metadata Terms (Dublin Core) ontology [22]. The information model provides categories for the representation of content as well as properties for relations to products, technical components, functions, target audience and the product lifecycle phases. The focus in the iiRDS information model is on metadata that describes information (information units). Additionally, the iiRDS ontology provides docking points for the integration of other ontologies, e.g. domain- or company-specific taxonomies of products or components. The following sections give a brief overview of the iiRDS information model and its most important elements.

#### Element Overview

The iiRDS information model consists of classes, properties and some predefined constants. Figure 3.9 depicts an overview of the iiRDS information model. Basically, `InformationUnits` are used to represent/reference single parts of content. `InformationUnits` are then annotated with `InformationTypes`.

The main elements of the iiRDS ontology and its corresponding purposes are:

- **InformationUnit:**  
Parent class for representing content elements like documents, topics, fragments or packages.
- **Document:**  
Subclass of `InformationUnit` that is a collection of information units.
- **Topic:**  
Subclass of `InformationUnit` that is a self contained chunk of information.

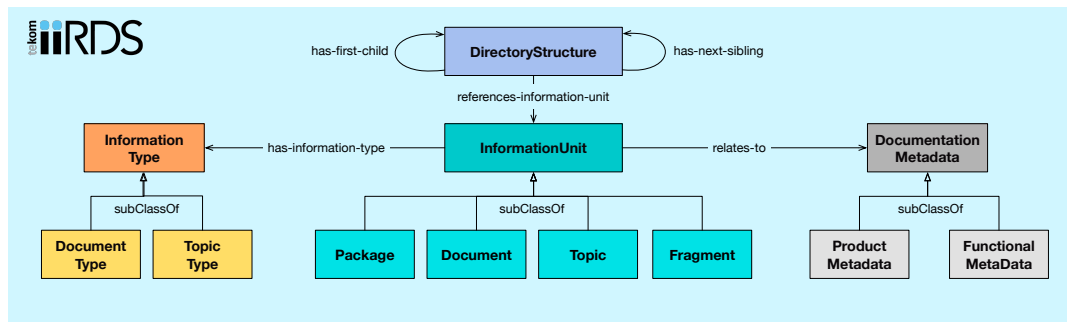


FIGURE 3.9 | Overview of the iiRDS information model.

- **Fragment:**  
Subclass of InformationUnit representing a small chunk of information that is not self contained, i.e. a reader requires additional context information.
- **Package:**  
Subclass of InformationUnit that is a collection of information units together with metadata and relations.
- **InformationType:**  
Specifies the type of content, e.g. document types, topic types, information subjects.
- **DocumentType:**  
Subclass of InformationType that defines the type of an information unit, e.g. repair instruction or assembly information.
- **TopicType:**  
Subclass of InformationType that defines the type of a topic, e.g. task, concept or learning.
- **DirectoryStructure:**  
Represents relations and hierarchies between information units.
- **Documentation-Metadata:**  
Parent class for all product-related metadata, functional metadata or docking points for taxonomies of products or components.
- **Rendition:**  
Represent references from information units to physical content/ source files.
- **FragmentSelector:**  
Constrain Renditions to certain fragments of the physical content.
- **RangeSelector:**  
Constrain Renditions to a specific range of the physical content.

The following sections describe the respective elements in more detail.

### Structural Elements

The iiRDS information model contains elements that facilitate the construction of structural representations. Such structures are intended to be used for navigation

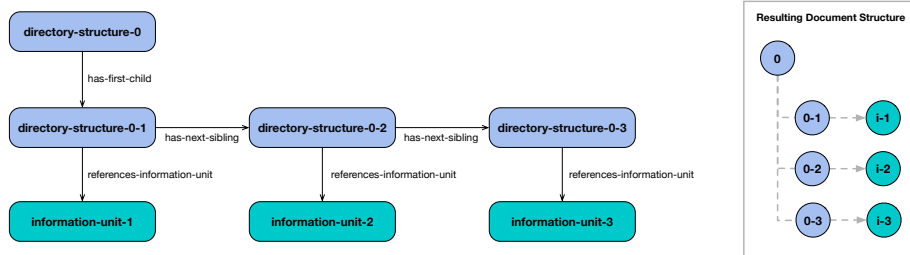


FIGURE 3.10 | Directory nodes for the representation of document structures.

purposes in dynamic documentation or to assemble a printable version of the documentation. Therefore, the iiRDS specification provides the `DirectoryNode` class with corresponding properties (`has-next-sibling` and `has-first-child`). `DirectoryNode` instances model navigation sequences and additional structures like hierarchies of `InformationUnits`. The actual `InformationUnit` is referenced from `DirectoryNode` instances using the `references-information-unit` property. This facilitates accessing context information of `InformationUnit` instances like siblings, predecessors or successor information units. Structures that have been assembled using `DirectoryNode` instances can be typed using the `has-directory-structure-type`. This facilitates the explicit definitions of document structures like tables of contents. An overview of document structure modeling using the iiRDS vocabulary is depicted in Figure 3.10.

### Rhetorical Elements

The iiRDS information model does not explicitly provide rhetorical elements. Instead, the specification provides a wide range of possibilities to define metadata. These metadata facilitates the inclusion of rhetorical aspects into the technical documentation. All metadata in the iiRDS specification is defined using the `DocumentationMetadata` class or respective subclasses for functional (`FunctionalMetadata`) or product (`ProductMetadata`) metadata. The rhetorical aspects are mainly encapsulated by `FunctionalMetadata`. Examples are `Event`, `Supplies`, `PlanningTime` or `Qualification`. Dedicated properties exist in the iiRDS specification for the annotation of `InformationUnit` instances with such metadata, e.g.:

- `relates-to-event`:  
e.g. error codes.
- `has-subject`:  
e.g. subject of the `InformationUnit`: Information Subjects: purpose of a chunk of information.
- `requires-supply`:  
e.g. tools, spare parts.
- `requires-qualification`:  
e.g. professional skills, training certificates, qualifications.
- `requires-time`:  
e.g. duration or interval of tasks.
- `relates-to-product-metadata`:  
e.g. product, component, product life-cycle phase, or product feature.

### Additional Metadata

The iiRDS specification also allows for the definition of additional metadata, e.g. for defining topics in terms of products, components or other relevant concepts of the underlying domain. As the specification focuses on technical documentation it contains some predefined metadata classes and singleton instances for this application scenario. The main classes for such metadata are `ProductMetadata` and its subclasses `Product`, `ProductVariant`, `ProductFeature`, and `Component`. The `ProductMetadata` class is a subclass of the more general `DocumentationMetadata` class and serves as docking point for external ontologies. This way, elements from external ontologies like company- or industry-specific taxonomies can be used as metadata for iiRDS `InformationUnit` instances. The idea is that products, assemblies and parts are modeled as stub concepts using the iiRDS vocabulary, i.e. the modeled elements do not represent actual physical products or components. Hierarchies of stub concepts can be created using the `has-component` property. Then, the modeled stub concepts are aligned/mapped to actual external ontologies using `rdfs:seeAlso`.

### TEKNO Extension: iiRDS

From a structural view, the iiRDS information model does not provide equivalents to the fine grained nano structures of the TEKNO ontology. However, there are correspondences to `tekno:MicroStructure` and `tekno:MacroStructure`, i.e. `Fragment` and `Topic/Document` and their subclasses respectively. The rhetorical aspects of the TEKNO ontology are covered by the `InformationType` construct. More precisely, the `TopicType` class of the iiRDS information model corresponds to TEKNO's information type. The iiRDS information does not natively support Core Documentation Entities. However, with the basic information of the information model and the aforementioned mappings to the TEKNO ontology, Core Documentation Entities can be deduced from technical documents in iiRDS format. Figure 3.11 shows the mapping between the iiRDS information model and the TEKNO ontology. Bold arrows between the two models indicate the mapping.

### Formats and Delivery

Information modeled according to the iiRDS specification needs to be packaged in a standardized structure — the iiRDS container. Basically, the iiRDS container is a directory structure with a single root directory and can be realized as ZIP archive, file system or code repository. Besides the actual documentation content (physical information units) an iiRDS container must contain certain meta information. This meta information must be defined in a file called `metadata.rdf` in a `META-INF` directory. The `metadata.rdf` contains all semantic information about the physical information units, like structural definitions (c.f. `DirectoryNode` instances), rhetorical information (c.f. `FunctionalMetadata`), and additional metadata (c.f. `ProductMetadata`).

### Tool Support

The iiRDS specification aims on two kinds of applications — iiRDS generators and iiRDS consumers. While iiRDS generators are intended to facilitate the creation of iiRDS compliant documentation, iiRDS consumers exploit the standardized format to deliver the documentation to the end user. Typical examples for iiRDS generators are content management systems (CMS). Consumers of iiRDS data are typically content delivery portals or apps.

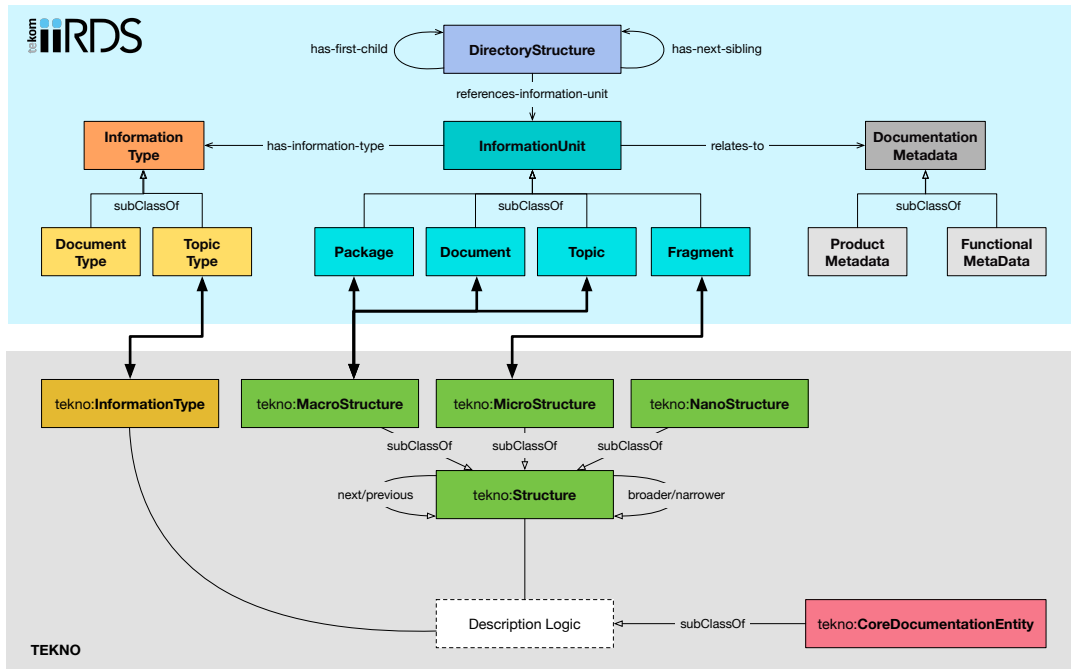


FIGURE 3.11 | Mapping of the iiRDS information model to the TEKNO ontology.

## Discussion

While technical documentation authored according to the iiRDS information model in general yields a 5-STAR rating in the maturity schema, there are minor shortcomings. Nano structures as required for the first maturity level are generally not available. Accessing such structures needs additional processing (e.g. custom tokenizers). Similarly, micro structures like paragraphs, headlines etc. are not necessarily available as the intended usage of the **Fragment** aims on isolating special paragraphs for reuse. The macro structures required from the third maturity level are given through different subclasses of **InformationType** and the dedicated structuring class **DirectoryStructure**. The third maturity level additionally requires modularization, which is perfectly met by the iiRDS information model through the key class **InformationUnit**. The fourth star requires information typing and identifiability for modules. Both requirements are fulfilled, as the **InformationType** class is a fundamental element in the iiRDS information model and iiRDS data is defined in RDF which requires URIs. The requirements of the fifth star are met through the **DocumentMetadata** class which enables the annotation of information units with elements of a product/component taxonomy. In summary, the iiRDS-Mod information model enables the creation of 5-STAR technical documentation with limitations with respect to nano (1-STAR) and micro structures (2-STAR). A shortcoming of the whole model is that Core Documentation Entities are not available. However, the conjunction of the iiRDS information model and the Core Documentation Entity expressions of the TEKNO ontology facilitates good accessibility. Table 3.7 summarizes this discussion according to the a 5-STAR rating schema.



1-STAR	2-STAR	3-STAR	4-STAR	5-STAR
				

TABLE 3.7 | iiRDS 5-STAR rating.

### 3.4.3 DITA

DITA (Darwin Information Typing Architecture) was first defined in the late 1990s by IBM [56]. In 2004 IBM handed DITA over to OASIS (Organization for the Advancement of Structured Information Standards). The current DITA specification version 1.2 comprises:

- **Language Definition:**  
XML DTDs and XML schemas defining the DITA language
- **Documentation:**  
A specification describing DITA concepts and a reference of all elements and attributes of the language.
- **DITA Open Toolkit:**  
Scripts and programs for validating and processing DITA source files.

DITA originated as a specification for technical documentation — especially software documentation. However today, the topic based approach of DITA is also applied in other areas like marketing, training or product information.

#### Information Model

The DITA information model is provided as a language definition in XML Doctype Definitions (DTD) and XML schemas. The information model provides dedicated XML elements for defining rhetorical aspects (specialized **topics**) as well as for the hierarchical structuring of the information (topic maps and content elements for topics). Additionally, the DITA information model allows for the specialization of DITA's predefined topics in order to meet domain- or customer-specific requirements. The following sections give a brief overview of the DITA information model and its most important elements.

#### Element Overview

Figure 3.12 depicts an overview of the DITA information model. Basically, **Topics** are instantiated in order to represent self-contained content elements. Each **Topic** instance is assembled from **Elements**. **Elements** are either applicable to all topics or constrained to certain topic types. Maps are used to sequence/structure single topics in a collection. In general DITA distinguishes the following categories of elements:

- **Topics:**  
Basic unit for representing content. Must have a title and optionally a body of content.

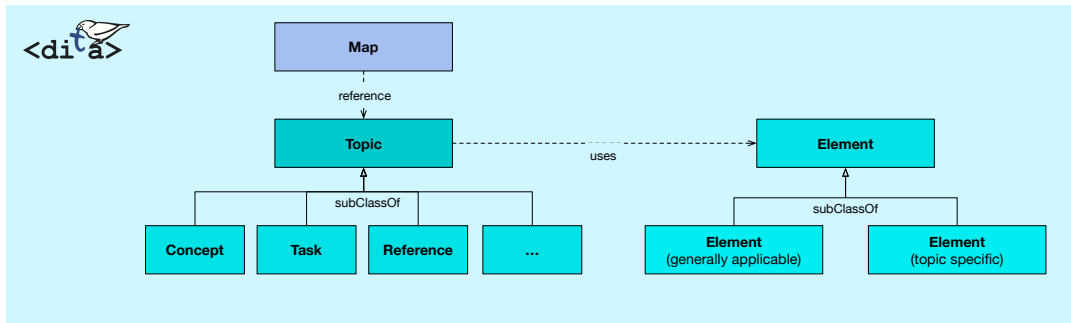


FIGURE 3.12 | Overview of the DITA information model.

- **Maps:**  
Documents that organize topics into structured collection of information. Specify hierarchies and relationships among topics.
- **Information typing:**  
Identify the types of topics, e.g. concepts, references, tasks.
- **Linking:**  
Links are used for defining maps, topic-to-topic navigation, cross references and reuse of content by reference.
- **Addressing**  
Relationships in DITA can either be defined directly (URI-based) or indirectly (key-based).
- **Content Reuse:**  
Mechanisms for reusing content fragments within topics or maps.
- **Conditional Processing:**  
DITAs form of applicability, i.e. processing is conditioned by attributes.
- **Constraints:**  
Constraints can be used to define allowed vocabulary, element or attribute types or other language elements.

The aforementioned elements are used to modularly define self-contained topics, interlink them and finally assemble (conditioned) publications using DITA maps, conditional processing, and different addressing mechanisms.

### Structural Elements

The basic units of information in DITA are topics. Although there are specializations of the general DITA topic, all of them have the same basic structure, i.e. a title and an optional body of content. The optional body of content is the main source for more fine-grained structural elements in DITA. These more fine-grained structural elements comprise:

- **Basic topic elements,**  
e.g. title, abstract, shortdescription
- **Body elements,**  
e.g. ordered (ol) and unordered lists (ul), section or definitions (dd, dl, dt).

- **Table elements**,  
e.g. `thead`, `tbody`, `row` or `entry`
- **Related link elements**,  
e.g. `link`, `linktext` or `linkinfo`

While the aforementioned language elements allow for a fine grained structural composition of contents, DITA maps facilitate the construction of hierarchical structures above topics. Therefore, DITA maps are used to build up hierarchies where each element is a referenced topic. Additionally, topic references in DITA maps can be conditioned/constrained to facilitate the assembly of different outputs/publications from one DITA map.

### Rhetorical Elements

DITA allows to categorize topics according to different reader questions, like "How...?" or "What is...?". Therefore, DITA supports specializations of the general `<topic>` element. Hence, from a rhetorical perspective the main language elements of the DITA specification are the specialized topic types. The DITA specification comes with a small set of predefined and well-established topic types, which are specializations of the main `<topic>`: `<concept>`, `<task>`, `<reference>`, and `<learningContent>`. DITA's specialization mechanism can be used to define an unbounded amount of additional topic types, which either specialize the most general type `<topic>` or refine the aforementioned subtypes.

### Core Documentation Entities

The DITA language specification also contains *Core Documentation Entities*, i.e. elements that combine structural and rhetorical aspects to clearly define the semantic of a certain piece of text. In DITA, these elements are tightly coupled to corresponding topic types, e.g. `<task>` topics are accompanied by certain task requirement elements like `<prereq>` or `<step>`. For the `<glossary>` topic, there type exist specialized glossary elements like `<glossentry>` or `<glossterm>`. Other specialized elements exist for the `<reference>` and the `<bookmap>` topic types. Additionally, the DITA language specification contains specialized elements for certain problem domains. Examples comprise programming elements, software elements, and user interface elements. A large amount of specialized elements exist for defining learning and training contents.

### Additional Metadata

Through specialized topic types and the availability of corresponding Core Documentation Entities DITA topics inherently contain a lot of semantics. Additionally, the language specification contains elements for the explicit definition of metadata. In general, DITA allows for the definition of metadata in topics (`<prolog>` element) and maps (`<topicmeta>` element).

Additionally, DITA provides indexing mechanisms which can also be used for metadata definitions. The indexing mechanisms are realized through dedicated classification elements that facilitate the definition of subject metadata. Therefore, DITA allows for the definition of subject scheme maps where a problem domain can be modeled in terms of subjects. The modeled subjects can then be used to classify topics with these predefined topics. The subject scheme map also allows for the definition of relations between single subject instances.

### TEKNO Extension: DITA

The DITA information model does not provide direct equivalents for rhetorical elements of the TEKNO ontology. The `tekno:InformationType` classes can not be mapped directly to elements of the DITA information model. Hence, the TEKNO extension for the DITA information model concentrates on mapping the structural elements. Basically, DITA topics are similar to `tekno:MacroStructures`. Some of DITA's topic specializations inherently define a `tekno:InformationType`, e.g. DITA's topic specialization `Concept` induce the information type `tekno:Description`. On the micro level DITA's dedicated XML elements are typical `tekno:MicroStructures`. XML elements that are constrained to certain topic specializations have Core Documentation Entity character. Figure 3.13 shows the mapping between the DITA information model and the TEKNO ontology. Bold arrows between the two models indicate the mapping.

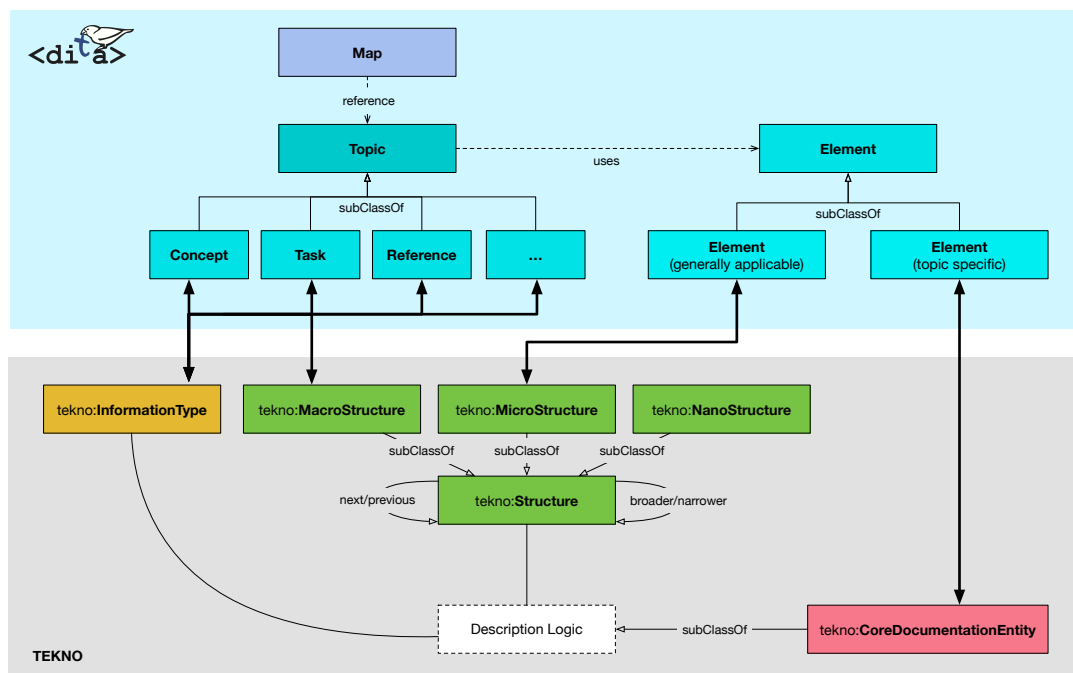


FIGURE 3.13 | Mapping of the DITA information model to the TEKNO ontology.

### Formats and Delivery

A technical documentation in DITA format basically is a collection of XML files. Each XML file represents a single topic in DITA. Additionally, a DITA map needs to be defined in order to combine or sequence the topics in a meaningful manner. Multiple DITA maps can be defined for different information products. A single DITA map can be constrained to adapt an information product to different audiences. The DITA source files (DITA XML files and corresponding DITA maps) are then processed by a DITA XML processor which exports the information to the desired output format, e.g. PDF or HTML. The formatting for the output medium can be customized using cascading or XSL stylesheets.

## Tool Support

DITA topics and maps can be created and edited with standard XML editors. Some XML editors have built in support for DITA and thus guide the author through the creation of a documentation. There also exist full fledged management systems for DITA sources with more elaborate functions for reuse, variables, and variant control. The content delivery of DITA-based documentation do not require specialized software. The DITA sources are processed and exported to standard formats like PDF or HTML content. However, specialized —merely proprietary— content delivery tools exists that exploit DITA’s language elements to provide better access to information.

## Discussion

In general, technical documentation authored according to the DITA information model fulfills huge parts of the 5-STAR maturity schema. Access to nano and micro structures as required for the first and second maturity level is available through XML elements. Depending on the usage of available elements this provides very targeted access to information pieces. The third maturity level requires modularization, which is one of the fundamental ideas of the DITA information model. Topics describe self-contained pieces of information which are intended for multiple use through maps (which are a possibility in DITA to define macro structure hierarchies). The fourth star requires information typing and identifiability for modules. While the identifiability of modules is ensured through unique identifiers of DITA topics, the information typing is questionable. Some predefined topic specializations can be mapped backwards to information types. An additional way to inject an information type into topics is to employ the mechanisms for defining additional metadata. The requirements of the fifth star are also met through the ability to define metadata in maps and topics which enables the annotation of macro structures with elements of an external ontology. In summary, the DITA information model enables the creation of 5-STAR technical documentation. The conjunction of the DITA information model and the Core Documentation Entity expressions of the TEKNO ontology facilitates good accessibility. An explicit possibility to define information types would further improve the handling of DITA topics. Table 3.8 summarizes this discussion according to the a 5-STAR rating schema.

1-STAR	2-STAR	3-STAR	4-STAR	5-STAR
				

TABLE 3.8 | DITA 5-STAR rating.

### 3.4.4 DocBook

DocBook [146] is a semantic markup language that has been developed since the early 1990s. The first ideas were discussed in respective usenet discussion groups in 1991. Then, the ideas were manifested and maintained by HaL, O’Reilly, Davenport and OASIS. Now, the DocBook Technical Committee at OASIS maintains the specification. The DocBook language is a structured markup that is mainly intended for hardware and software documentation. Documentation according to the DocBook

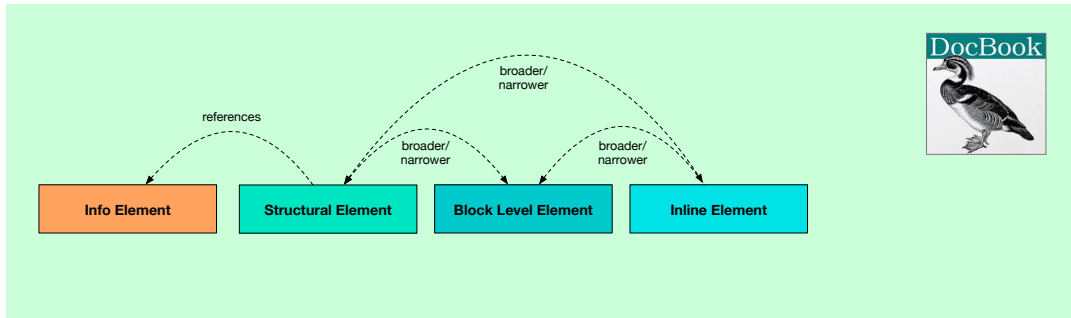


FIGURE 3.14 | Overview of the DITA information model.

specification is written in XML (formerly also in SGML) and can be validated against RELAX NG [42] and XML schema or Doctype Definitions (DTDs). Nowadays, DocBook gets also applied to other fields than technical documentation, although it is not primarily intended for such additional use cases.

### Information Model

The DocBook information model is primarily provided as RELAX NG schema. The information model also exists as XML schema and Doctype Definition, which both get derived from the RELAX NG schema. The DocBook Publishers extension allows for the usage of other (de facto) standards like the DCMI Metadata Terms (Dublin Core) ontology [22]. The DocBook information model provides a vast amount of specific XML elements. The focus of the information model is on the semantic representation of block and inline elements of (technical) documentation. Additionally, the DocBook specification allows for the customization of the underlying schemas in order to represent domain- or customer-specific elements. The following sections give a brief overview of the DocBook information model and its most important elements.

### Element Overview

In general DocBook distinguishes the following three categories of elements (see Figure 3.14):

- **Structural elements:**  
Used to (hierarchically) structure a document.
- **Block-Level elements:**  
Used to structure the content of a concrete information unit.
- **Inline elements:**  
Used to semantically represent concrete information elements.

Additionally, DocBook provides elements for meta-information, e.g. elements for representing metadata using the Dublin Core vocabulary.

### Structural Elements

Using the three basic element types technical documentation is assembled in a tree structure in DocBook. The structural elements of the DocBook specification typically form the upper levels of such a tree structure. Books are divided into divisions, divisions into components, and components into sections respectively. The most important structural elements on the respective levels are:

- **set:**  
Collection of at least one book that is nestable with other sets.
- **book:**  
Collection of chapters, articles, and/or parts.
- **part:**  
Collection of at least one chapters; nestable with other parts.
- **article:**  
Collection of block-level elements or chapters.
- **chapter:**  
Collection of block-level elements.

Below the structural elements either block-level or inline elements form the lower levels of the DocBook tree structure. Block-level elements usually divide concrete chapters or sections into smaller portions. Typical examples for block-level elements are the **paragraph** or **list** elements. Inside of these blocks, inline elements can be used to add fine-grained semantics or elements for highlighting (**emph**) or navigational purposes (**link**) to the content.

### Rhetorical Elements

The DocBook specification does not provide dedicated rhetorical elements. However, the **info** element that serves as a wrapper for information about a component or block might be used to add some rhetorical information to blocks or components. Appropriate children elements of the **info** element for this purpose comprise **keywordset**, **termset** or **subjectset**.

### Core Documentation Entities

Although the DocBook specification does not provide dedicated rhetorical elements it comes with a large amount of merely structural elements with a predefined semantic. In fact, the DocBook specification aims to provide a semantic markup language for technical documentation and thus the element collection contains a lot of fine-grained elements for representing important aspects of technical documents. These elements are mainly block or inline elements.

However, as these elements have a merely structural character it is necessary to augment such content with respective **info** elements in order to meet the idea of Core Documentation Entities. Listing 3.1 shows a procedure as a typical example. A **subject** element was added to the **info** section to define the semantics of the procedure more precisely, i.e., from the example data it can be conducted that it is a repair sequence.

```

1 <article xmlns='http://docbook.org/ns/docbook'>
2 <info>
3   <subjectset>
4     <subject>Repair</subject>
5   </subjectset>
6 </info>
7 <title>Example procedure</title>
8 <procedure>
9   <title>An Example Procedure</title>
10  <step>
11    <para>A Step</para>
12  </step>
13  <step>
14    <para>Another Step</para>
15    <substeps>
16      <step>
17        <para>Substeps can be nested indefinitely</para>
18      </step>
19    </substeps>
20  </step>
21  <step>
22    <para>A Final Step</para>
23  </step>
24 </procedure>
25 </article>

```

LISTING 3.1 | DocBook procedure augmented with rhetorical aspects.

### Additional Metadata

Basically, the DocBook specification allows for the addition of metadata to most kinds of elements. Therefore, the aforementioned `info` element or the dedicated elements within the DocBook Publishers extension of the Dublin Core can be used directly. Listing 3.2 shows an updated example where the `info` element not only describes a rhetorical aspect (Repair) of the article but also the component in focus (Gear Box 0815).

```

1 <article xmlns='http://docbook.org/ns/docbook'>
2 <info>
3   <subjectset>
4     <subject>Repair</subject>
5     <subject>Gear Box 0815</subject>
6   </subjectset>
7 </info>
8 ...
9 </article>

```

LISTING 3.2 | DocBook info with metadata for components.

### TEKNO Extension: DocBook

In general the DocBook information model provides similar elements to the TEKNO ontology. From a structural perspective, a structural element in DocBook gets assembled from block-level and inline elements. This corresponds to the assembly of a `tekno:MacroStructure` according to TEKNO. In TEKNO `tekno:MicroStructure` and `tekno:NanoStructure` instances are used for the assembly respectively. DocBook's `info` element corresponds to the `tekno:InformationType` in the TEKNO ontology.



However, unlike TEKNO's information type its usage is strictly constrained to structural elements. Core Documentation Entities can be derived from DocBook structural elements that contain an info element. The derivation of Core Documentation Entities is however not intended by DocBook and thus requires the aforementioned mapping of the DocBook information model to the TEKNO ontology. Figure 3.15 shows the mapping between the DocBook information model and the TEKNO ontology. Bold arrows between the two models indicate the mapping.

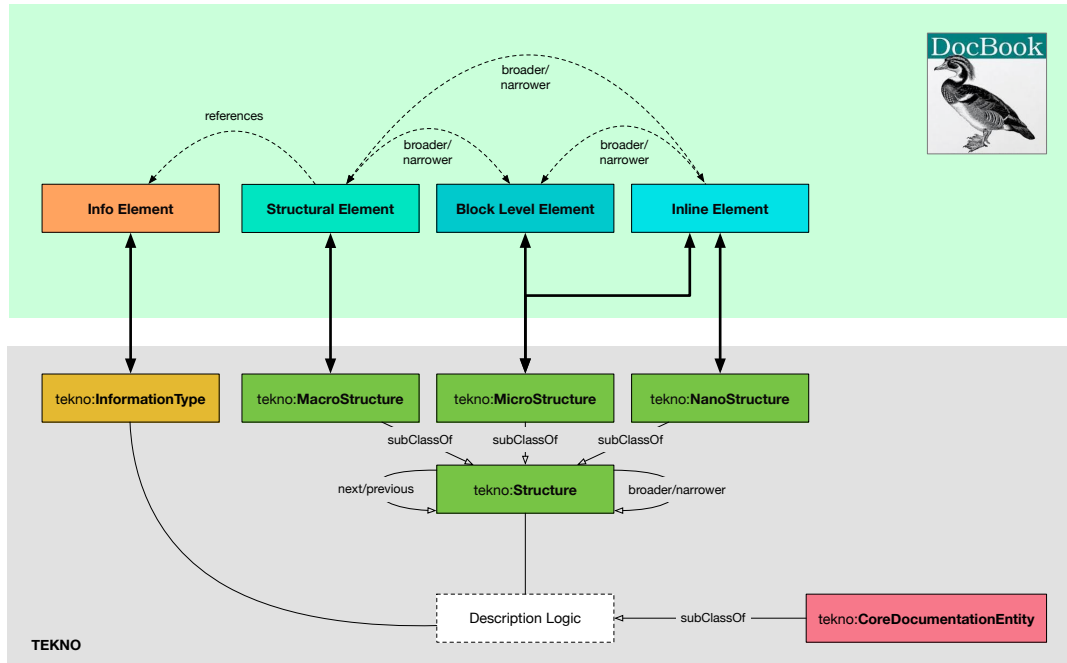


FIGURE 3.15 | Mapping of the DocBook information model to the TEKNO ontology.

## Formats and Delivery

Technical documentation written according to the DocBook specification must be serialized in XML format. The content can be separated to different XML files, e.g. for editing chapters in single files. An include mechanism ensures that the separated source files can be assembled to a complete publication.

For the DocBook XML language a variety of XSL stylesheets exist. These stylesheets can be used to transform the DocBook XML source files to a variety of output formats like HTML or XSL-FO (PDF). The transformation comprises the automatic generation of certain document structures like table of contents, glossaries or other indexes. The XML language can be customized to meet domain- or customer-specific requirements. However, the customization requires the editing of the standard RELAX-NG schema which might break tool support relying on the standard.

## Tool Support

The creation of a documentation according to the DocBook specification does not require a dedicated editor. The documentation can be created and edited with a standard XML editor. Some editors come with a predefined support for DocBook XML.

For the transformation of DocBook XML files to the supported output formats XSL stylesheets can be applied using a standard XSL processor. Standard XML validators can be used to validate DocBook source files against the RELAX-NG/XML schema or the Doctype Definition (DTD).

### Discussion

The DocBook information model provides —from a structural perspective— almost perfect mapping abilities to the TEKNO ontology. For the core elements of the TEKNO ontology, a one to one mapping to DocBook elements is possible. Although, the definition of rhetorical information is limited and not standardized. Nevertheless, the DocBook information model fulfills the requirements of the 5-STAR maturity schema. Access to nano and micro structures as required for the first and second maturity level is available through XML elements. Depending on the usage of available elements this provides very targeted access to information pieces. The third maturity level requires modularization. While the DocBook information model in general provides the required functionality it depends on the actual authoring if a modularization happens. A document written according to the DocBook information model can either be composed from multiple files (modules) or written as a single file. The fourth star requires information typing and identifiability for modules. All structures in DocBook documents are identifiable to basic XML features. Information Typing can be realized through the usage of DocBook’s info element. The requirements of the fifth star are also met through the ability of defining metadata enables the annotation of macro structures with elements of an external ontology. In summary, the DocBook information model enables the creation of 5-STAR technical documentation. The conjunction of the DocBook information model and the Core Documentation Entity expressions of the TEKNO ontology facilitates good accessibility. A strict requirement to author DocBook contents in a modularized way could further improve the handling of this information model. A standardized way for the definition of rhetorical aspects would ensure that generated documentation supports rhetorical filtering. Table 3.9 summarizes this discussion according to the a 5-STAR rating schema.






1-STAR	2-STAR	3-STAR	4-STAR	5-STAR
				

TABLE 3.9 | DocBook 5-STAR rating.

### 3.4.5 S1000D

S1000D is an international specification for authoring and procuring technical documentation. It was originally developed by the European Association of Aerospace Industries (AECMA). Nowadays, it is maintained by the S1000D Steering Committee, which mainly consists of the AeroSpace and Defence Industries Association of Europe (ASD), the Aerospace Industries Association (AIA) and the Air Transport Association (ATA). The specification provides a guideline for the standardized creation of technical documentations and supports the country and organization independent information exchange through uniform documentation standards. The primary target groups of the specification are the aerospace industry and military organizations.

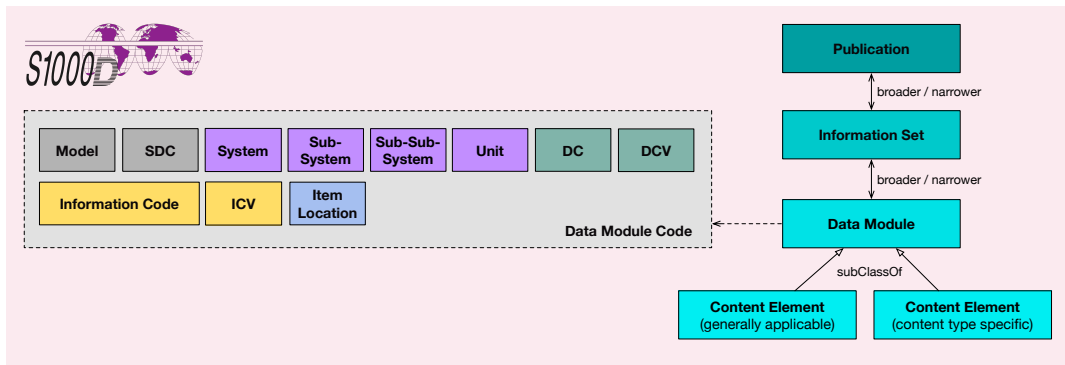


FIGURE 3.16 | Overview of the S1000D information model.

However, the S1000D specification is also used for the creation of technical documentation in other (civil) domains like construction or the ship industry. In general, S1000D is the most recognized international standard for technical publications.

### Information Model

The S1000D information model is primarily provided as SGML and XML Document Type Definitions (DTD). The information model also exists as XML schema. The information model contains different Document Type Definitions for the various information types, e.g. specialized DTDs for descriptive or procedural information. A vital element of the information model is the Standard Number System (SNS) that is used for the encoding of components and procedures. A key idea of the S1000D information model is modularization of information. Therefore, S1000D uses so called data modules, i.e. small pieces of technical information. Data modules as well as illustrations get alphanumeric codes – the Data Module Code (DMC) or Illustration Control Number (ICN) respectively. These codes are the basis for the usage of a Common Source Database (CSDB) as a centralized storage and management medium of data modules and illustrations. Additionally, the data module codes and illustration control numbers simplify the information exchange among project members or institutions. The S1000D information model also contains descriptions of publication types (e.g. operating instructions, repair manuals or troubleshooting documents) and output formats (e.g. interactive documentations or printed publications).

### Element Overview

Figure 3.16 depicts an overview of the S1000D information model. The main element of the S1000D specification is the data module. A data module is a small, self-contained and reusable piece of technical documentation. In S1000D they form the smallest information unit within a technical publication.

Closely coupled with the data module is the data module code (DMC) that encodes the hierarchical breakdown of a system. The data module code is a generic classification system for the complete documentation that also encodes other data like location or information codes. The data module code also allows for the determination of relations between documented entities, e.g. sub component/part of relations.

The data modules can be organized in so called information sets which in turn can be used to assemble publications (macro level structuring). The S1000D specification contains descriptions for the different publication types.

Additionally, the S1000D specification contains comprehensive descriptions of elements that can be used within the data modules (micro level structuring). These content elements of the S1000D specification facilitate the semantic description of technical content. For each information type it is clearly defined where specific elements (e.g. safety notes) have to appear and how additional metadata can be defined, e.g. required skill-levels or preliminary and subsequent work.

### Structural Elements

From a structural view the S1000D information model provides elements that facilitate structuring technical information encapsulated in data modules on a micro and macro level. On the macro level the S1000D information model provides so called information sets. An information set provides information for a predefined scope in form of appropriately arranged data modules. The S1000D information model comes with some predefined information sets, e.g.:

- Crew/Operator information
- Maintenance information
- Wiring data
- Illustrated Parts data
- Service bulletins
- Training

These information sets serve as a basis for the assembly of publications. A publication can either be assembled from subsets of one information set or as a super set of different information sets. In some cases it might be necessary to group data modules, because their contents are similar. An example for such a scenario is a maintenance action that achieves the same goal but needs different detailed procedures depending on the product configuration or environment conditions. Therefore, so called container data modules exist that allow for explicitly stating alternatives.

On the micro level the S1000D information model provides a variety of dedicated XML elements for different types of data module content sections. These elements allow for a detailed semantic description of technical information. Some of these XML elements can be applied to all types of content sections, e.g. references, lists, tables, figures with hotspots or general text elements and thus have a purely structural character. The usage of other elements is limited to certain types of content sections and are highly specialized and thus have Core Documentation Entity character (see Section 3.4.5).

### Rhetorical Elements

The rhetorical elements of the S1000D elements are closely coupled with the aforementioned information sets. For each data module an information code (IC) has to be defined as part of the data module code (DMC). This information code facilitates the identification of the rhetorical aspect of a single data module.

Table 3.10 shows the top level elements of the information codes of the S1000D information model. These top level elements are specialized on the subsequent levels and thus form a taxonomy of information codes.

Information Code	Description
000	Function, data for plans and description
100	Operation
200	Servicing
300	Examinations, tests and checks
400	Fault reports and isolation procedures
500	Disconnect, remove and disassemble procedures
600	Repairs and locally make procedures and data
700	Assemble, install and connect procedures
800	Package, handling, storage and transportation
900	Miscellaneous
C00	Computer systems, software and data

TABLE 3.10 | S1000D information codes and descriptions.

### Core Documentation Entities

The S1000D information model describes a lot of dedicated XML elements. Some of them are generally applicable (see Section 3.4.5) whilst others are limited to certain types of data module content sections. As the type of the data module content sections is closely coupled with the rhetorical elements (see Section 3.4.5) elements with a constrained applicability usually have Core Documentation character. The following list shows some typical examples of data module content types and samples of respective dedicated XML elements with Core Documentation Entity character:

- **Fault information:**  
faultIsolationProcedure, isolationStep
- **Maintenance check list:**  
checkListProcedure, checkListItem
- **Wiring data:**  
wire, fromEquip, toEquip

Most of these dedicated elements are group elements, i.e. they contain other, usually generally applicable text elements for the concrete definition of the content. This limits the machine-interpretability of the content but still simplifies the processing and the subsequent access of the corresponding information.

### Additional Metadata

In the S1000D information model, each data module has a unique data module code. The data module code encodes a lot of information which in turn can be interpreted as classifying metadata. Figure 3.17 gives an overview of the elements of the data module code.

The complete data module code identifies a single data module. When broken into its parts the data module code allows to deduct a lot of classifying metadata for the respective data module. The following elements can usually be mapped to corresponding ontologies/taxonomies.

- **Model:**  
Indicates the complete model/system of the documentation.

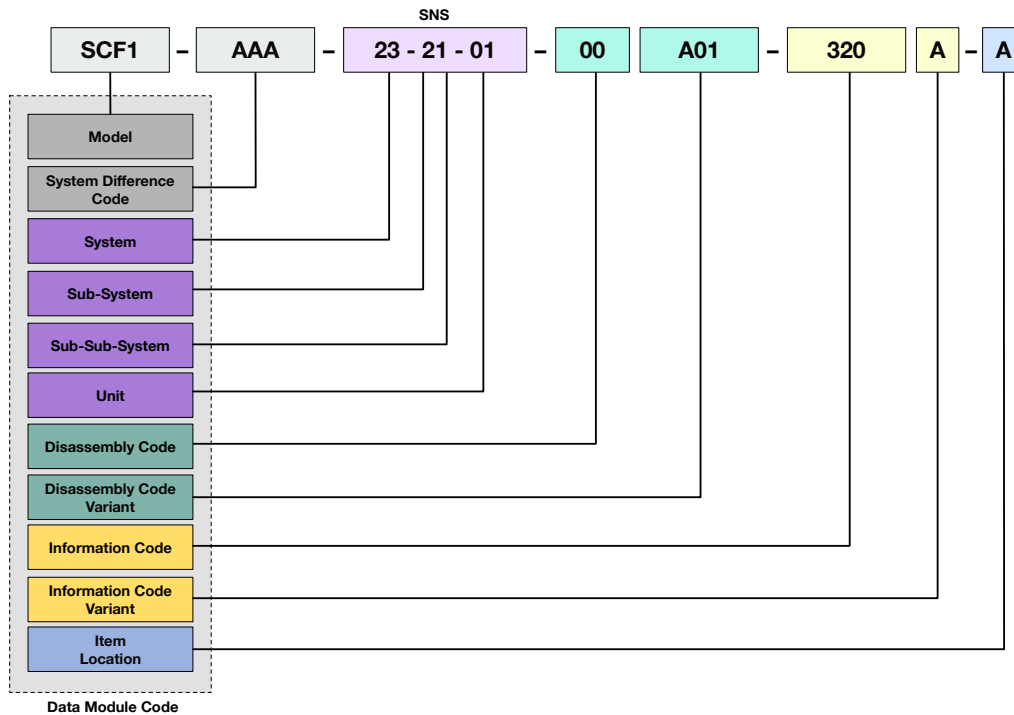


FIGURE 3.17 | S1000D data module code.

- **System Difference Code:**  
Indicates alternative versions of the model.
- **System:**  
Identifies the system described by the data module.
- **Subsystem/Sub-subsystem:**  
The further breakdown of the system.
- **Unit/Assembly Code:**  
Breakdown beyond System/SubSystem/Sub-Subsystem level.
- **Information Code/Variant:**  
Indicates the type of information.
- **Item Location:**  
The location of the unit.

Additionally, the S1000D information model allows for the addition of metadata like information in the `content` part of data modules. Therefore, a dedicated **applicability** element exists which can handle human-readable as well as machine-interpretable values. Usually the applicability is used to define variants of machines to which the information described in the data module applies.

### TEKNO Extension: S1000D

In general the S1000D information model provides similar elements to the TEKNO ontology and thus enables an almost complete mapping of both models. From a structural perspective a data module assembled from content elements according to the S1000D information model corresponds to a `tekno:MacroStructure` instance with

`tekno:MicroStructure` instances respectively. Instances of S1000D information sets or publications correspond to higher level macro structures in the TEKNO ontology. In S1000D the information code and information code variant is encoded in the data module code. These elements are equivalent to the `tekno:InformationType`. Core Documentation Entities can either be declared directly using respective S1000D content elements constrained to certain content types or can be derived from content elements, the assigned information code of the enclosing data module and the respective expressions in the TEKNO ontology. The latter requires the aforementioned mapping of the S1000D information model to the TEKNO ontology. Figure 3.18 shows the mapping between the S1000D information model and the TEKNO ontology. Bold arrows between the two models indicate the mapping.

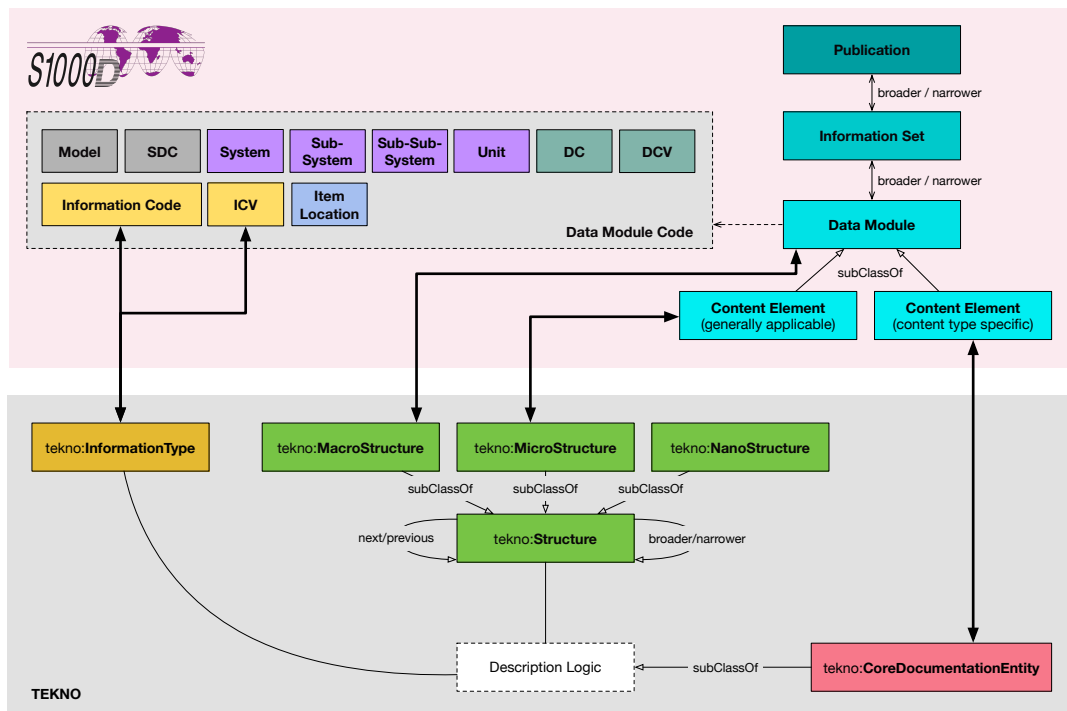


FIGURE 3.18 | Mapping of the S1000D information model to the TEKNO ontology.

## Formats and Delivery

Technical Documentation written according to the S1000D specification must be in XML/SGML format. The content of the complete technical documentation is spread over different XML files, where each XML file represents a single data module. Information sets and publications get assembled with dedicated XML files which set the respective data modules in a reasonable sequence. Additionally, the S1000D specification requires illustrations to be in CGM (Computer Graphics Metafile) format, 3D animations can be defined using VRML. The data modules in XML files are usually not stored as files on the file system but as entries in a Common Source Database.

## Tool Support

The S1000D specification is non-proprietary, i.e. it allows for neutral delivery and management. Hence, a variety of tools exist that support the creation, editing and management of S1000D data modules and publications. Additionally, the S1000D

specification provides Document Type Definitions and XML schemas that can be used to validate the XML content of data modules.

### Discussion

In general, technical documentation authored according to the S1000D information model fulfills huge parts of the 5-STAR maturity schema. Access to nano and micro structures as required for the first and second maturity level are available through dedicated XML elements. Depending on the usage of available elements this provides very targeted access to information pieces. The third maturity level requires modularization, which is one of the fundamental ideas of the S1000D information model. Each data module describes a self-contained piece of information which is intended for multiple use through information sets or publications (which are a possibility in S1000D to define macro structure hierarchies). The fourth star requires information typing and identifiability for modules. The identifiability of data modules is ensured through its unique data module code. The data module code also contains an information code that defines the information type of the respective data module. Although S1000D encodes a lot of information in the data module it is hardly possible to define additional metadata annotations from external ontologies. The S1000D information model is a fully integrated information management framework and requires relevant concepts to be defined in a S1000D compatible way, i.e. machine components are usually encoded using the standard numbering system. The only way to directly reference concepts from an external ontology is the abuse of content elements like **remarks**. Indirect references to concepts from external ontologies can be realized by mapping them to reasonable parts of data module codes, e.g. the standard numbering system. This, however, is not part of the S1000D specification and requires the support of the consuming information system. Though, the basic requirements of the fifth star are met, as data modules inherently encode their subject. In summary, the S1000D information model enables the creation of de-facto 5-STAR technical documentation. The conjunction of the S1000D information model and the Core Documentation Entity expressions of the TEKNO ontology facilitates good accessibility. An explicit possibility to reference concepts from external ontologies would further improve the handling of S1000D-based technical documentation. Table 3.11 summarizes this discussion according to the a 5-STAR rating schema.


1-STAR	2-STAR	3-STAR	4-STAR	5-STAR
				

TABLE 3.11 | S1000D 5-STAR rating.



### 3.5 Summary and Contributions

The latest developments in the fields of information systems require technical documentation to be more and more semantically prepared. Section 3.2 discussed the demand of a possibility to quickly assess the maturity of existing technical documents and the underlying information models. As there is no standardized and easy to apply schema a novel 5-STAR maturity schema has been introduced. The maturity schema has five levels and a star is added on each level if documentation meets the respective requirements. This way structural accessibility on different granularity levels (1-STAR, 2-STAR, 3-STAR), modularization (3-STAR), identifiability and information typing (4-STAR) and linkability (5-STAR) can quickly be assessed.

Building upon the 5-STAR maturity schema TEKNO as a novel abstracting meta ontology for technical documents has been introduced in Section 3.3. TEKNO provides a simple but powerful ontological vocabulary to semantically represent structural and rhetorical elements of technical documentation. Through its meta character it can be easily aligned to existing information models. Such alignments usually yield considerable improvements regarding accessibility. This is especially emphasized through a catalog of so called Core Documentation Entities that is part of the TEKNO ontology. Core Documentation Entities combine existing structural and rhetorical elements to deduct elements that carry strong technical knowledge. Examples for such Core Documentation Entities are repair instructions, component overviews, and measurement tables.

Level	Description	PI-Mod	iiRDS	DITA	Doc-Book	S1000D
1	Nano Structures	★	☆	★	★	★
2	Micro Structures	★	☆	★	★	★
3	Macro Structures	★/	★	★	★/	★
4	Information Types	★	★	★/	★/	★
5	Annotations	★	★	★	★	★/

TABLE 3.12 | Summary of 5-STAR ratings for the information models PI-Mod, iiRDS, DITA, DocBook, and S1000D.

Section 3.4 gave an overview of established information models for the authoring of technical documents, namely: PI-Mod, iiRDS, DITA, DocBook, and S1000D. For each information model the most important structural and rhetorical elements have been identified. Additionally, a mapping of the TEKNO ontology to the respective information model has been described. All information models have been assessed according to the 5-STAR maturity schema. The results of the single assessments have been discussed, also with respect to the aligned TEKNO ontology, please refer to Table 3.12 for an overview. The fact that none of the analyzed information models receives a 5-STAR rating emphasizes the demand of an abstracting meta ontology like TEKNO.



## Chapter 4

# Semantification of Technical Documents

*An investment in knowledge pays the best interest.*

*Benjamin Franklin*

---

## 4.1 Overview

The previous chapter discussed different kinds of semantics for technical documents and gave an overview of existing information models. Although a variety of structured information models exist large amounts of technical documents still reside in proprietary or legacy formats. Such formats usually lack accessibility and linkability and thus inherently limit their usage in state-of-the-art information systems. This chapter describes a semantification process that is split into five steps that build upon each other and transforms legacy documents into an accessible and linkable representation. The five steps are aligned with the maturity model that was introduced in Section 3.2:

1. **Document Layout Analysis:**

The first semantification step is concerned with fundamental document layout analysis, which aims on recovering pages, contiguous text blocks, sentences, and single tokens. This is a challenging task especially for scans and documents in PDF format. For a detailed description please refer to Section 4.3.

2. **Logical Document Structure Recovery:**

The second semantification step aims on recovering the logical document structure, which especially affects the classification of contiguous text blocks. For a detailed description please refer to Section 4.4.

3. **Macro Structure Recovery and Deduplication:**

The third semantification step exploits classified text blocks like headlines in order to recover macro structure hierarchies. Macro structure hierarchies correspond to chapters, sections and subsections. This is the basis for the modularization and deduplication of large documents. For a detailed description please refer to Section 4.5.

4. **Automatic Document Classification:**

The deduplicated modules of the third semantification step are automatically classified according to rhetorical classes. These rhetorical classes correspond to information types like “repair”, “diagnosis” or “description”. The classification of modules according to information types enables rhetorical filtering in respective information systems. For a detailed description please refer to Section 4.6.

5. **Subject Analysis and Indexing:**

The final semantification step aims on enabling problem-oriented and targeted search. Therefore, modules must be annotated with concepts from a domain ontology. Assuming that such a domain ontology describes functions and components of a machine problem-oriented and targeted access to relevant modules becomes possible. For a detailed description please refer to Section 4.7.

Each semantification step introduces an information system use case and thus motivates its realization with respect to an actual application scenario. Building upon formal problem descriptions relevant approaches from literature and novel techniques

are thoroughly presented and discussed for each semantification step. The semantification steps have already been applied in a series of industrial projects. The experiences gained in these projects are manifested in practical representations that are given at the end of each section.

## 4.2 Preparatory Steps

This section describes preparatory steps that are especially necessary in industrial projects. Since real-world projects often aim on the semantification of large corpora of legacy documents with a fixed or limited budget a prioritization might be necessary. Additionally, a data cleaning step aims on quickly identifying documents that are not in a processable state.

### 4.2.1 Data Selection

Basically, the 5-STAR semantification approach allows for the batch processing of legacy documents. However, in real-world projects tailoring single steps is necessary in order to yield decent results. As this tailoring is especially based on expert knowledge it easily gets cost-intensive. Therefore, a prioritization of the legacy documents might be necessary. In the data selection phase the documents contained in the corpus are therefore grouped into disjunct priority lists (see Figure 4.1).

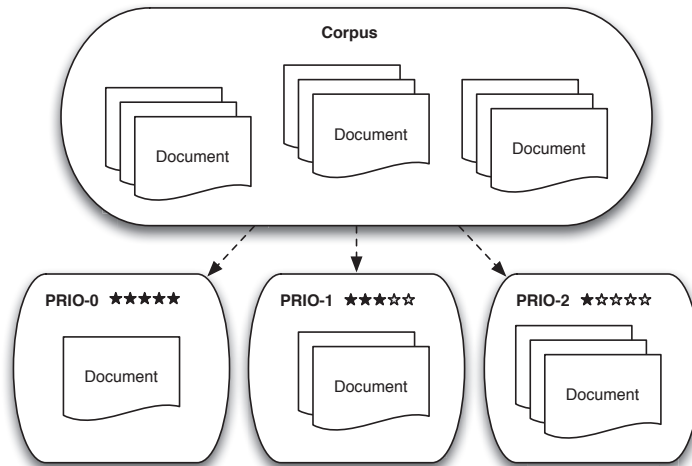


FIGURE 4.1 | Prioritization of documents on priority lists.

**Definition 4.2.1** (Priority List). Let  $K$  be the corpus of technical documents and let  $D_i \in K$  a document. Then a priority list  $P_b \subseteq K$  is defined as a set of documents

$$P_b = \{D_1, \dots, D_n\}$$

so that

$$\forall_b \forall_c P_b \cap P_c = \emptyset \text{ with } b \neq c \text{ and } \bigcup_b P_b = K.$$

The creation of the priority lists is usually a rather manual task as corporate aspects such as market penetration of the corresponding product or customer satisfaction needs to be taken into account. Workshops with different stakeholders for the definition of the priority lists are indispensable. It is not necessary to define priority

lists on the document level. These lists are typically defined on the machine and document type level, e.g. including repair manuals of a specific machine type into a specific priority list.

### 4.2.2 Data Cleaning

The documents selected for semantification might not be in a processable state, i.e., documents may be damaged, encrypted or use unsupported encodings. Therefore, the semantification process contains an explicit data cleaning step, that checks the processability of the selected documents.

**Definition 4.2.2** (Processable Documents). Let  $K$  be the complete corpus of technical documents and let  $D_i \in K$  be a document. Then the function  $pcorp : K \rightarrow K_p$  extracts all processable documents from the corpus by examining each document  $D_i$  using the function  $pdoc : D \rightarrow BOOLEAN$ , so that  $K_p \subseteq K$ .

The function  $pdoc$  checks every document for processability, which for instance can be realized by using heuristics. The documents identified as not processable

$$K_{np} = K \setminus K_p$$

need manual review before they can be processed. Depending on their importance the review might also result in an exclusion of the respective documents.

## 4.3 1-STAR

This section describes the first of five semantification steps. The 1-STAR semantification is the fundamental basis for all succeeding semantification steps. It aims on providing access to the text of pages, blocks, sentences and single tokens. This is a challenging task when confronted with scanned texts or documents in PDF format.

### 4.3.1 Information Systems Use Case

Technical documents fulfilling the first maturity level are usually employed in text-based information systems. Text-based information systems usually require the full plain text of documents in order to build a textual index. Therefore, document texts are extracted from scanned documents or documents in PDF format. The extracted text then usually gets tokenized. Then, the text is prepared for indexing by removing stop words and applying stemming techniques to single tokens. Finally, an inverted index from tokens to document pages gets assembled under consideration of relevance metrics. This allows text-based information systems to provide page-wise access to documents even if they are originally provided as scanned images or in PDF format. The inverted index is the basis for text-based retrieval mechanisms that consume user keyword queries and determine relevant search results.

### 4.3.2 Problem Description

The 5-STAR maturity schema requires 1-STAR technical documents to be available in an electronic format. This electronic format must at least provide access to the following structures:

- **Pages:** a single page of a document.
- **Blocks:** contiguous text within a single page that can usually visually separated from other text elements, e.g. headlines or paragraphs.
- **Texts:** contiguous tokens that form a self-contained element, e.g. headline text or sentences.
- **Tokens:** single words.

The reconstruction of such structures is part of the more general research topics of *Optical Character Recognition* [142] and *Document Layout Analysis* [150] (also *Page Layout Analysis*). Optical Character Recognition is the conversion of images of typed, handwritten or printed text into machine-encoded text. Document Layout Analysis approaches usually try to segment content in text and non-text zones. Elaborated approaches also determine high level classes for text zones like paragraph, heading or page number and add relationship information to them.

Applications of both the Document Layout Analysis problem and Optical Character Recognition can be found for instance in historical book recognition where electronic text is extracted from scanned, ancient books. In the context of technical documentation, Document Layout Analysis is a topic when source data of documents is either not available or exists in a highly unstructured or proprietary format. Typical examples are documents that originally have been written using typewriters or documents in formats like PDF that basically describe the arrangement of single glyphs.

Access to the aforementioned document structures requires the transformation of image- or glyph-based documents to zoned electronic text represented as micro

(blocks, texts) and nano structures (tokens). Therefore, a document's content needs to be segmented into disjunct zones, analyzed, and recovered. The resulting electronic description of technical documents can then be exploited to either consume the textual content or to employ more elaborate analysis and recovery tasks.

The remainder of this section is structured as follows: In Section 4.3.3 some basic information about skew estimation is given, which plays a vital role for image-based documents. Subsequently Section 4.3.3 introduces zoning mechanisms that separate homogeneous text and non-text regions. It continues with the explanation of the recovery of nano structures (tokens) and the subsequent recovery of micro structures for page segmentation purposes. Established 1-STAR electronic formats for the representation of basic document structures are presented in Section 4.3.4. Section 4.3.5 gives practical recommendations for applying Logical Document Structure Recovery to technical documents.

### 4.3.3 Document Layout Analysis

The goal of the *Document Layout Analysis* problem is the estimation of a hierarchical representation of a document, i.e., the representation of the document on different levels of detail under consideration of spatial relationships [35]. For instance, a document can be represented on a high level by enumerating its pages. Increasing the level of detail yields access to blocks, lines, and finally tokens. Additionally, Document Layout Analysis approaches ensure that elements detected on different levels are hierarchically connected to each other. The following sections describe in detail the required steps for reaching these goals.

#### Skew Estimation

Documents that originally have been written on typewriters are usually only available as scanned images. Thus, the very first step of Document Layout Analysis is skew estimation. Skew estimation approaches determine the deviation of the document orientation angle from the horizontal and vertical direction [35]. Figure 4.2 depicts the skew estimation process.

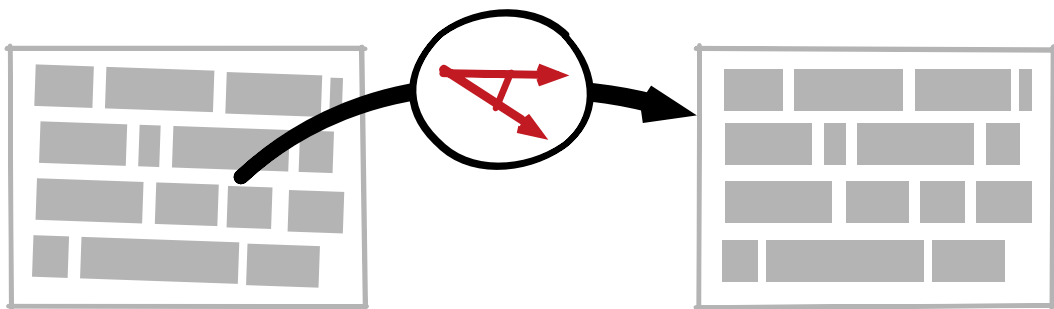


FIGURE 4.2 | Skew estimation for a single page.

Common approaches for skew estimation have been surveyed by Cattoni et al. [35]. The skew estimation problem is well-established research topic. Among others, common approaches are based on the following techniques:

- **Analysis of projection profiles:**  
Text is assumed to be arranged along parallel straight lines [35]. Then a projection profile is computed, an objective function over the skew angle defined and optimized [35].



- **Hough transformation:**  
A feature extraction technique that uses a certain voting procedure to find imperfect instances of geometrical objects.
- **Nearest neighbor clustering:**  
Characters are supposed to appear in a line and are aligned close to each other [35]. Mutual distances and spatial relationships between objects are exploited to estimate the document skew [35].
- **Correlation between lines:**  
Assumes that deskewed text appears in homogeneous horizontal structures [35]. Hence vertical deviations are used to estimate a document's skew [35].

Other approaches comprise gradient analysis, analysis of the fourier spectrum, morphological transformations and subspace line detection. For more information please refer to the survey work of Cattoni et al. [35].

### Text/Non-Text Zoning

Documents that are available in PDF or image format first need to undergo a text/non-text zoning process. This step separates textual from non-textual regions. This is an important aspect for both PDF and image-based documents. Contents of image-based documents can be filtered to allow for a targeted employment of subsequent OCR tasks which usually is beneficial with respect to the expected results. The PDF format on the other hand does not provide a logical representation for encapsulated text. However, neither image-based documents nor documents in PDF format give direct access to separate text and non-text zones.

Considering image-based/scanned documents it is obvious that the separation of text and non-text zones requires further processing. While documents in PDF format give the impression that access to textual element is easily possible, the PDF format in fact aims primarily at the preservation of the intended look of documents. Hence, documents in PDF format usually do not provide access to a correct logical representation of textual contents [162]. Instead, it provides all kinds of content in forms of object streams that need to be separated. Common objects in PDF streams are: text, path objects for the definition of vectorial elements, and external objects like raster images. While the stream elements can easily be partitioned according to their type (see Figure 4.3) problems arise when document components are composed from different types of stream elements. This is especially true for vectorial images that can be assembled from path and text elements.

This makes the employment of dedicated zoning algorithms necessary for both image-based documents and documents in PDF format. Established methods for separating documents into text and non-text zones are based on XY-Cut algorithms. XY-Cut algorithms [144, 136] usually transform a document into a whitespace density graph where peaks indicate whitespace lines (horizontal or vertical). The peaks are then used top-down to cut a document page into smaller blocks. Whitespace information is easily available for image-based documents. For documents in PDF format the information has to be computed from dimensions available for single page and stream elements of different types.

### Nano Structure Reconstruction

Given clearly separated text and non-text zones the subsequent step in Document Layout Analysis is the recovery of nano structures aka tokens. Tokens are not directly

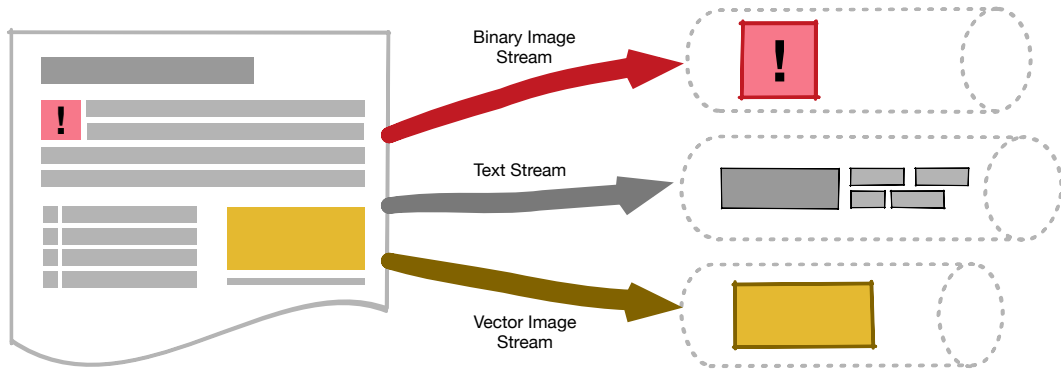


FIGURE 4.3 | Separation of text and non-text zones.

available from the estimated text zones. For image-based documents the recovery of single tokens is usually covered by a dedicated Optical Character Recognition step that relies on different approaches for tokenizing character streams. For documents in PDF format each text zone is represented as a set of textual stream elements. It is undefined whether a single textual stream element corresponds to a single character, a partial word, a complete word, a partial line or another arbitrary partition of the text zone [162].

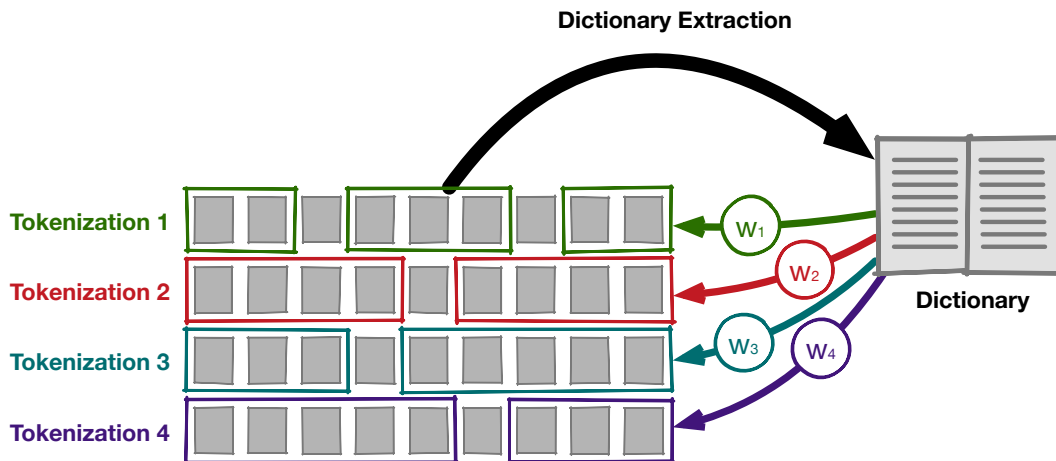


FIGURE 4.4 | Tokenization for Nano Structure Recovery.

In literature different approaches for the tokenization of textual streams from PDF documents can be found. Dejean et al. [52] proposed an industry proven approach that combines spatial/geometrical and statistical/dictionary information (see Figure 4.4). First, an initial tokenization is computed solely based on geometrical/spatial information. Using this initial tokenization a weighted dictionary is built where the weight of each token corresponds to its document frequency. A high frequency yields a higher weight than a lower weight. Then, for every line all possible tokenizations with respect to the generated dictionary are computed. A Viterbi-style algorithm is then used to select the most appropriate tokenization, i.e., the tokenization with the best overall-weighting.

### Micro Structure Segmentation

The results of the text and non-text zoning step do not necessarily correspond to micro structures like single headlines, paragraphs, and tables. Instead, text zones might also contain blocks of nested micro structures that need additional segmentation. Thus, in a particular micro structure segmentation step the text zones undergo an additional division into smaller pieces (see Figure 4.5). The result is another segmentation of the document in (potentially more granular) homogeneous regions [35]. While the goal of this step is to produce homogeneous blocks of maximum size the main feature under consideration is the spacing among different regions [35].

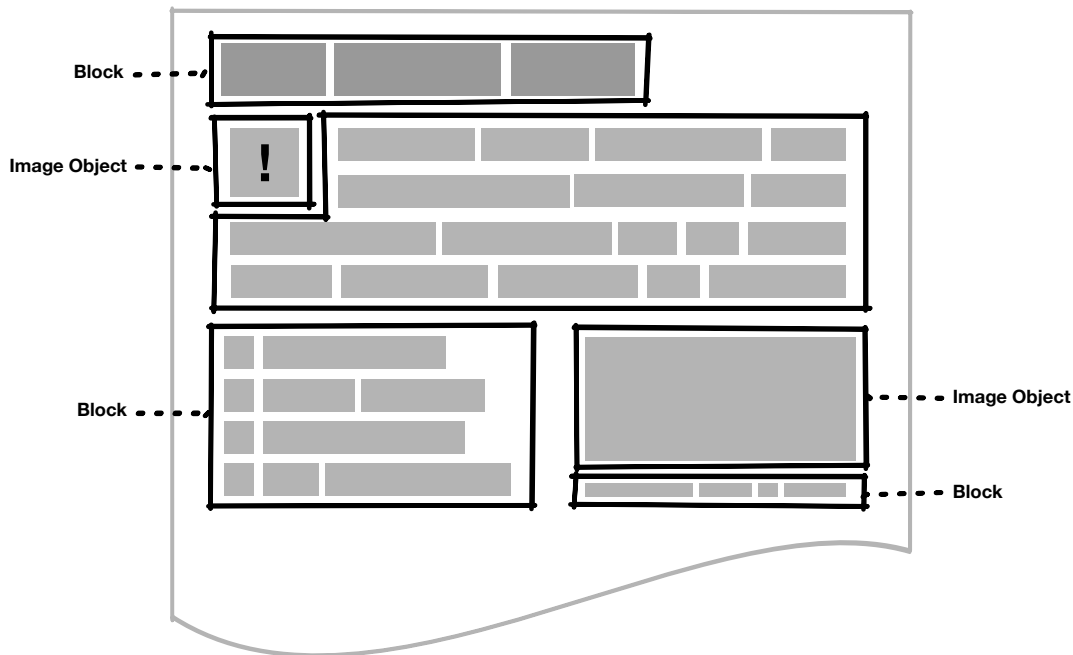


FIGURE 4.5 | Segmentation for Micro Structure Recovery.

Cattoni et al. [35] surveyed approaches for the *Page Decomposition* problem, which is another term for the aforementioned micro structure segmentation. The surveyed approaches come mainly from two sub disciplines called *Text Segmentation* and *Page Segmentation*. The surveyed approaches rely amongst others on the following methods:

- Connected Component Analysis:** Textual stream elements are transformed to vertices of a graph. The vertices contain additional information (features like spatial or font information) which is exploited by specialized heuristics that determine connected neighbors (i.e. similar elements). Ramakrishnan et al. [162] presented an approach that also considers multi-column layouts.
- Projection Profile Methods:** The projection profile of a document image (see skew estimation) is exploited to segment the text using recursive XY-Cut algorithms. The basis for the segmentation are peaks and valleys in the projection profile [35].

- **Smearing-based Techniques:**

Operate on sequences of pixels which correspond to rows or columns of an image-based document. The pixels are binary (0,1) encoded and get transformed by certain rules. This way, areas of 0's and 1's result. These areas are considered to be a micro structure on threshold basis. Smearing based algorithms are considered to be fast and easy to implement and are thus popular in respective systems [35].

- **Analysis of the background structure:**

An extension of smearing based approaches that do not work on a single sequence of pixels but a two-dimensional representation of pixels. Pixels are replaced by an index value. This is the basis for the computation of a hierarchical tree structure that is exploited for the computation of the resulting segmentation [35].

- **Texture based or local analysis:**

A document image gets processed on pixel-level. For each pixel a probability is estimated that indicates the pixel to belong to a word block. Using probability thresholds, connected blocks of words can be detected [35].

#### 4.3.4 Electronic Formats

The Document Layout Analysis described in Section 4.3.3 yields access to basic document components. After the analysis steps, document components are available on different levels of detail. Starting with elements representing complete pages a successive increase of detail allows accessing the page content separated in homogeneous blocks of text. The text within these blocks is then represented in lines consisting of single tokens. Various export formats exist for the delivery of the recovered structures. After introducing the basic requirements of such delivery formats, the following sections will give a non-exhaustive overview of some of these formats.

#### Requirements

The presented approaches for Document Layout Analysis often have domain-specific target applications. Therefore, standardized and/or broadly accepted formats for the delivery of analysis results do not exist [156]. In this work, the Document Layout Analysis problem is embedded in a larger Semantification problem. Hence, the ongoing process of semantifying technical documents results in some requirements regarding document representation formats. The requirements affect elements, attributes/features for these elements and possibilities to express the logical layout.

Focusing on elements that must be provided by the electronic format the baseline comprises regions/blocks, contiguous text (lines) and single words. From the feature/attribute perspective for all textual elements information about their font, reading direction, text, and background color as well as applied text decorations (italics, bold, ...) must be available. Information about the logical layout of the document must at least provide possibilities to describe linear relations between elements, like next, previous, broader or narrower relations. Possibilities for grouping and nesting elements yield additional accessibility benefits.

### lapdf-text

Ramakrishnan et al. [162] developed the open source utility lapdf-text<sup>1</sup>. While the tool primarily aims on extracting *layout-aware* full text from PDFs, it supports the complete Document Layout Analysis task. It also provides an XML-based representation of its analysis results. Although the output is not standardized it does meet the basic requirements for a 1-STAR electronic format.

The lapdf-text XML format provides amongst others the following elements: text blocks (chunk) and tokens (word). For textual elements the following features are available: font-name, font-size, height, width, and the position on the page. There is no explicit way to represent linear logical relationships.

Listing 4.1 shows an example of a lapdf-text XML document. The example document contains several blocks which roughly correspond to lines. The content of these text blocks can be accessed on the token level. For each element detailed information about its position and style are available.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <Document>
3   <Page x1="34" y1="25" x2="566" y2="803"
4     chunkCount="..." pageNumber="1" wordCount="..." >
5   <Chunk x1="56" y1="25" x2="154" y2="39">
6     <Word x1="56" y1="25" x2="154" y2="39"
7       font="ArialMT" style="font-size:12pt">
8       Service Manual</Word>
9     </Chunk>
10    <Chunk x1="183" y1="67" x2="439" y2="84">
11      <Word x1="183" y1="67" x2="439" y2="84"
12        font="Helvetica - Bold" style="font-size:16pt">
13        Index</Word>
14      </Chunk>
15    <Chunk x1="56" y1="152" x2="520" y2="222">
16      <Word x1="56" y1="152" x2="404" y2="166"
17        font="ArialMT" style="font-size:12pt">
18        Foo</Word>
19      <Word x1="408" y1="152" x2="520" y2="166"
20        font="ArialMT" style="font-size:12pt">
21        Bar</Word>
22      <Word x1="396" y1="166" x2="520" y2="180"
23        font="ArialMT" style="font-size:12pt">
24        Baz</Word>
25    </Chunk>
26  </Page>
27 </Document>

```

LISTING 4.1 | An example of the lapdf-text electronic format.

### pdf2xml

Dejean et al. [52] developed the open source and industry-proven utility pdf2xml<sup>2</sup> that supports the Document Layout Analysis task. The utility comes with its own XML-based export format. Although it is not standardized it meets the requirements for a 1-STAR electronic format.

The pdf2xml XML format provides amongst others the following elements: typed blocks (text, image, ...), text (lines), and tokens (words). For textual elements the following features are available: font-name, font-color, font-size, rotation, angle, text

<sup>1</sup><https://github.com/BMKEG/lapdf-text>

<sup>2</sup><https://github.com/kermitt2/pdf2xml>

decorations, height, width, and the position on the page. There is no explicit way to represent linear logical relationships. Though, grouping and nesting of elements is possible.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <DOCUMENT>
3 <METADATA>
4 </METADATA>
5 <PAGE width="420" height="595" number="1" id="p1">
6 <BLOCK id="p1_b1">
7 <TEXT width="225.105" height="17.175" id="p1_t1"
8   x="124.14" y="52.4794">
9   <TOKEN sid="p1_s3" id="p1_w1" font-name="IMOGHB+Helvetica - Bold"
10     serif="yes" fixed-width="yes" bold="yes" italic="no"
11     font-size="15" font-color="#000000" rotation="0" angle="0"
12     x="124.14" y="52.4794" base="66.3844" width="225.105"
13     height="17.175">Foo</TOKEN>
14 </TEXT>
15 </BLOCK>
16 <BLOCK id="p1_b2">
17 <TEXT width="87.21" height="32.0142" id="p1_t2"
18   x="193.08" y="84.1455">
19   <TOKEN sid="p1_s4" id="p1_w2" font-name="imoghb+helvetica - bold"
20     serif="yes" fixed-width="yes" bold="yes" italic="no"
21     font-size="27.96" font-color="#000000" rotation="0" angle="0"
22     x="193.08" y="84.1455" base="110.064" width="48.259"
23     height="32.0142">Bar</TOKEN>
24   <TOKEN sid="p1_s5" id="p1_w3" font-name="imoghb+helvetica - bold"
25     serif="yes" fixed-width="yes" bold="yes" italic="no"
26     font-size="27.96" font-color="#000000" rotation="0" angle="0"
27     x="249.171" y="84.1455" base="110.064" width="31.1195"
28     height="32.0142">Baz</TOKEN>
29 </TEXT>
30 </BLOCK>
31 </PAGE>
32 </DOCUMENT>

```

LISTING 4.2 | An example of the pdf2xml electronic format.

Listing 4.2 shows an example of a pdf2xml XML document. The example document contains several text blocks. The content of the text block can be accessed on line (text) and token level. For each element detailed information about its position and style are available.

## hOCR

The hOCR format [28] aims at adapting the HTML/XHTML format together with CSS. Therefore, HTML and CSS is used for representing typographic markup. Additional information is embedded using the facilities of standard HTML.

The hOCR format allows all standard HTML and CSS markup elements to provide a structural description of the document. The authors claim that HTML is —due to its wide usage— best suited to handle typographic and linguistic phenomena across a variety of languages [28]. Additionally, the hOCR format provides logical markup that is able to express logical relations between HTML elements independent of the actual rendering on the page. This way hOCR, provides constructs to represent blocks on different levels of detail as well as lines. Additionally, elements for representing images, formulas, separators, and noise are available.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <html>
3 <head>
4 <title>Example</title>
5 <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
6 </head>
7 <body>
8 <div class="ocr_page" title="image alice_1.png; bbox 0 0 2488 3507">
9 <h3>
10 <span class="ocr_line" title="bbox 467 525 1386 588">
11 Example
12 </span>
13 </h3>
14 <p class="ocr_par">
15 <span class="ocr_line" title="bbox 533 888 2077 946">
16 Foo, Bar
17 </span>
18 </p>
19 <p class="ocr_par">
20 <span class="ocr_line" title="bbox 525 2627 2068 2684">
21 Baz
22 </span>
23 <span class="ocr_line" title="bbox 449 2687 2065 2743">
24 qux, quux
25 </span>
26 </p>
27 </div>
28 </body>
29 </html>
```

LISTING 4.3 | An example of the hOCR electronic format.

Listing 4.3 shows an example of a document in hOCR format. The standard HTML document is enhanced with additional information using class and title attributes. The example document represents a page with three block elements. A complete description of the hOCR standard is available as *living standard*<sup>3</sup>. A variety of tools for validating and processing documents in hOCR format is freely available<sup>4</sup>.

## METS and ALTO

The Metadata Encoding and Transmission Standard (METS) [48] and the Analyzed Layout and Text Object (ALTO) are XML-based open standards that are hosted by the Library of Congress and maintained by respective editorial boards. The basic standard is ALTO, which aims at representing the content and layout of single pages. The description contains styles, layout, and block information. In conjunction with METS, the embedding of different kinds of metadata and the description of the logical structure becomes possible. However, the two types of information are separated. An ALTO XML file describes mainly the content of a document while a METS XML file supplies additional descriptive information.

<sup>3</sup><http://kba.cloud/hocr-spec/1.2/>

<sup>4</sup><https://github.com/tmbdev/hocr-tools>

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
2 <alto>
3   <Styles>
4     <TextStyle ID="font0" FONTFAMILY="Courier New" FONTSIZE="9">
5   </Styles>
6   <Layout>
7     <Page ID="Page1" PHYSICAL_IMG_NR="1" HEIGHT="6053" WIDTH="4234">
8     <PrintSpace HEIGHT="5469" WIDTH="3522" VPOS="0" HPOS="-5">
9       <Illustration ID="1_8" HEIGHT="344" WIDTH="158" VPOS="0" HPOS="-5">
10        <Shape>
11          <Polygon POINTS="88,2 240,2 240,344 88,344 88,2"/>
12        </Shape>
13      </Illustration>
14      <TextBlock ID="1_9" HEIGHT="47" WIDTH="160" VPOS="179"
15        HPOS="3019" language="de" STYLEREFS="font0">
16        <Shape>
17          <Polygon POINTS="3108,228 3268,228 3268,272 3108,272 3108,228"/>
18        </Shape>
19        <TextLine HEIGHT="37" WIDTH="160" VPOS="186" HPOS="3019">
20          <String STYLE="bold" WG="1." CONTENT="Foo" HEIGHT="37"
21            WIDTH="160" VPOS="186" HPOS="3019"/>
22        </TextLine>
23      </TextBlock>
24    </PrintSpace>
25  </Page>
26 </Layout>
27 </alto>

```

LISTING 4.4 | An example of the ALTO electronic format.

The ALTO XML format provides amongst others the following elements: typed blocks (text, image, ...), lines and words. For textual elements the following features are available: font, color information, text decorations, height, width, position on the page. Information about the logical layout must be defined using an accompanying METS XML document. A METS XML document therefore contains a `structmap` element that is able to group and nest references to actual text blocks of an ALTO XML document.

Listing 4.4 shows an example of an ALTO XML document<sup>5</sup>. The example document contains a graphic and a text block. The content of the text block can be accessed on line and token (string) level. For each element, detailed information about its position and style is available.

A variety of software supporting the creation of ALTO and METS files is freely available<sup>6</sup>. Additionally XML schema files exist for ALTO XML<sup>7</sup> and METS XML<sup>8</sup> that allow for the validation of generated files.

## PageXML

PageXML [156] is an XML-based document representation format that records image characteristics, layout structure and page content. Besides its expressiveness it allows to store intermediate processing results for the different phases of the Document Layout Analysis task. Hence, intermediate results of the process can be referenced and evaluated.

<sup>5</sup>Taken from the official ALTO GitHub repository, available at: <https://altotml.github.io/>

<sup>6</sup><https://github.com/altotml/documentation/wiki/Software>

<sup>7</sup><http://www.loc.gov/standards/alto/v3/alto-3-1.xsd>

<sup>8</sup><http://www.loc.gov/standards/mets/mets.xsd>



The PageXML format provides the following elements: typed regions (text, image, line drawing, graphic, table, chart, separator, maths, noise and frame), text, lines, words, and glyphs. For textual elements, the following features are available: language, script, font, reading direction, text color, background color, text decorations and type. Information about the logical layout can be defined using linear relations. Additionally, elements can be grouped and nested. PageXML also provides support for defining layers.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <PcGts>
3 <Metadata>
4 <Creator>Sebastian Furth</Creator>
5 <Created>2018-02-22T03:13:37</Created>
6 <LastChange>2018-02-22T03:13:37</LastChange>
7 </Metadata>
8 <Page imageFilename="SamplePage.png" imageHeight="800"
9 imageWidth="500">
10 <ReadingOrder>
11 <OrderedGroup id="ro357564684568544579089">
12 <RegionRefIndexed regionRef="r0" index="0"/>
13 <RegionRefIndexed regionRef="r1" index="1"/>
14 <RegionRefIndexed regionRef="r2" index="2"/>
15 </OrderedGroup>
16 </ReadingOrder>
17 <TextRegion>
18 <TextLine id="r0">
19 <Coords points="25,30 25,55 235,55 235,30"/>
20 <TextEquiv><Unicode>Foo</Unicode></TextEquiv>
21 </TextLine>
22 </TextRegion>
23 <GraphicRegion id="r2">
24 <Coords points="430,60 430,450 765,450 765,60"/>
25 </GraphicRegion>
26 <TextRegion>
27 <TextLine id="r1">
28 <Coords points="25,310 25,430 400,430 400,310"/>
29 <TextEquiv><Unicode>Bar</Unicode></TextEquiv>
30 </TextLine>
31 </TextRegion>
32 </Page>
33 </PcGts>

```

LISTING 4.5 | An example of the PageXML electronic format.

Listing 4.5 shows a PageXML example document<sup>9</sup>. The example document contains two text regions and a graphics block. It additionally shows the definition of the reading order for the defined blocks.

For exporting Document Layout Analysis results to PageXML, libraries for Java and C++ are freely available<sup>10</sup>. Additionally, an XML schema<sup>11</sup> is available that allows for the validation of exported documents.

<sup>9</sup>Adapted from the PageXML example document available under:  
<http://www.primaresearch.org/schema/PAGE/gts/pagecontent/2017-07-15/Simple%20PAGE%20XML%20Example.pdf>

<sup>10</sup><http://www.primaresearch.org/tools/PAGELibraries>

<sup>11</sup><http://schema.primaresearch.org/PAGE/gts/pagecontent/2017-07-15/pagecontent.xsd>

### 4.3.5 Practical Recommendations

Section 4.3.3 explained the theoretical foundations of the 1-STAR semantification step, which aims on reconstructing basic document structures and giving access to elements like pages, contiguous blocks of text, and single tokens. Section 4.3.4 presented some electronic formats that support the representation of the recovery results. This section gives practical recommendations for 1-STAR semantification tasks.

1. **Pre-classify and organize documents:**

The theoretical foundations of the Document Layout Analysis Task introduced in Section 4.3.3 show that vital steps of the analysis process rely on background knowledge. Hence, the respective approaches yield the best results when they get tailored/configured appropriately for a set of similar documents. For this reason, it is highly recommended to pre-classify the document corpus and to subsequently organize the documents such that a tailored/configured processing for each sub-corpus becomes possible.

2. **Choose appropriate tools:**

A variety of tools that support the Document Layout Analysis task or parts of it exist. However, it is hardly possible to identify a tool that is a general recommendation. The main reason is that tools have usually been developed for a certain application scenario. Hence, they normally yield decent results in the respective scenario while they rather underperform under other conditions. Therefore, the recommendation is to prefer tools that give different ways of tailoring, configuration and customizing. Open source utilities that allow for a knowledge-based tailoring/configuration are usually good candidates for the shortlist.

3. **Check readability of documents:**

Practical experience show that although a document has a certain format it is not necessarily readable. This is especially true for documents in PDF format. Various versions and profiles exist for the PDF standard that do not have comparable tool support. Another problem is document encryption that is a standard feature of PDF documents but not well supported by tools. Hence, a Document Layout Analysis tool should be able to check the readability of documents without performing a complete analysis.

4. **Carefully consider fail-fast and soft-fail:**

Corpora of technical documents often comprise several thousands of files. Hence, the analysis phase usually takes some time. While processing a big amount of unstructured data it is obvious that processing can fail for some documents. It is also possible that the results do not meet the expectations. Therefore, tools should support soft-fails, i.e. continue the processing of the corpus even if single documents fail. Additionally, a tool should allow early insights into single results in order to stop a processing run that obviously does not meet the expectations.

5. **Choose export format:**

Plenty of export formats exist and standardization has not become a big topic in the area of Document Layout Analysis. Hence, tools should be preferred that support the desired output format. If this is not possible the expressivity of a tool's output format should allow for an easy transformation. The consideration should be primarily based on the subsequent processing, as for example the following semantification steps need a certain baseline of information.

### 4.3.6 Related Work

This chapter presented the 1-STAR semantification process and broke it down to the Document Layout Analysis problem. Although this chapter focused on processing steps that are primarily necessary for improving the accessibility of technical documents, the presented approaches are not limited to this kind of documents. However, a lot of related approaches exist for similar problems in other domains. Reul et al. [166] presented LAREX which is an open source tool for Document Layout Analysis and subsequent region extraction on early printed books. LAREX exports its results in PageXML format. More general utilities supporting the Document Layout Analysis task are Tesseract [182], OCRopus [29], SCRIBO [115], and Agora [163]. Additionally, plenty of proprietary tools exist, e.g. Aletheia<sup>12</sup>, or the products of the ABBYY family<sup>13</sup>.

---

<sup>12</sup><http://www.primaresearch.org/tools/Aletheia>

<sup>13</sup><https://www.abbyy.com/finereader/>

## 4.4 2-STAR<sup>14</sup>

### 4.4.1 Information Systems Use Case

1-STAR technical documents can be used to build text-based inverted indexes that are the fundamental basis for textual information systems. However, 1-STAR documents do not give access to special micro structures like headlines, headers, or footers. Such structures, however, can be exploited in order to guide and improve the indexing task, e.g. by preferring tokens that appear in such special structures. The availability of such micro structures also enable targeted information extraction. Observations show, that in technical documents structures like headlines, image captions or image legends contain valuable knowledge that can be exploited for the population of technical ontologies. Typical examples are ontologies that describe the structural composition of machines in forms of assemblies and components.

### 4.4.2 Problem Description

The 5-STAR maturity schema requires 2-STAR technical documents to provide access to basic document structures like paragraphs, headlines, lists, or tables. Section 3.3.1 introduced these structures as *micro structures*. The result of the 1-STAR semantification process is a set of unordered and untyped micro structures. The recovered micro structures correspond to headlines on different levels, paragraphs, tables, and other homogeneous blocks of text. However, the accessibility of the recovered structures is not yet optimal as types and reading order remain unknown. Therefore, approaches from *Logical Document Structure Recovery and Analysis* (also *Logical Layout Analysis*) aim on typing micro structures (see Figure 4.6) and determining logical relationships among them. The resulting information can then be exploited for further semantification steps or other application scenarios like reconstructing the originally intended reading order. The actual relationships between blocks usually depend on the underlying document model. The structural elements of the TEKNO ontology (see Section 3.3.1) describe such a document model by providing classes for the representation of single structures and properties for the interconnection of them.

The recovery of the logical structures of documents is often a rather knowledge-intensive task. Due to varying document layouts, the involvement of domain- or project-specific background knowledge usually increases results significantly. Pre-classifying documents and the subsequent processing with specialized models usually further increase the result quality [35]. However, due to the knowledge-intensive character of Logical Document Structure Recovery and Analysis, generally applicable approaches are rather rare [35]. Cattoni et al. [35] surveyed approaches for the Logical Document Structure Recovery and Analysis problem. The underlying methods comprise different kinds of knowledge-based systems ranging from classical Expert Systems to various types of Machine Learning approaches. Cattoni et al. [35] report that they observe the best results on restricted domains. Typical examples for such restricted domains are scientific publications or technical documents where document layouts are usually constant for certain periods of time.

Logical Document Structure Recovery and Analysis is a well-established research topic in the field of scientific literature analysis. It is mainly concerned with the

<sup>14</sup>The contents of this section and its subsections are significantly extended and revised versions of the following published article: Sebastian Furth, Maximilian Schirm, Volker Belli, and Joachim Baumeister. "TEKNO: Preparing Legacy Technical Documents for Semantic Information Systems." Natural Language Processing and Information Systems: 22nd International Conference on Applications of Natural Language to Information Systems, NLDB 2017.

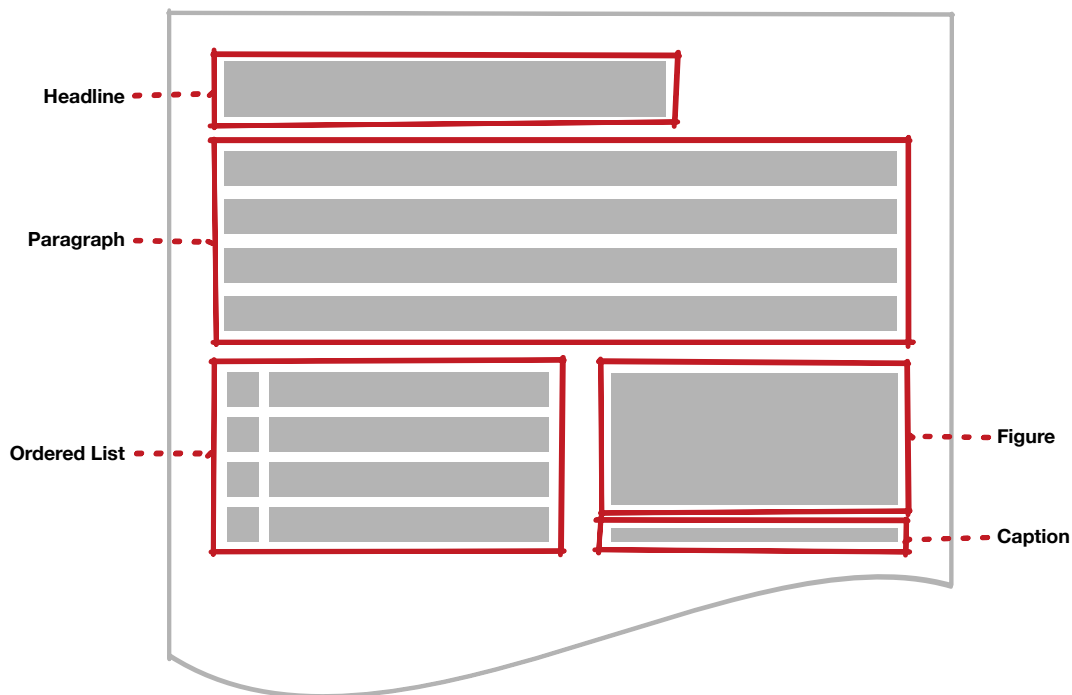


FIGURE 4.6 | Micro structures.

detection of hierarchies of logical components, for instance titles, authors, affiliations, abstracts, or sections; see Mao et al. [127]. Applications can be found in the area of bioinformatics, where scientific articles are analyzed in order to index them in meta-data based biomedical information systems like PubMed [131]. In the context of semantic information systems, document structure recovery is a major topic as it builds the fundamental basis for the integration of legacy data in these systems. The integration of such legacy data requires the transformation of existing documents to a semantic representation of modular information units with clearly defined meanings. This can only be realized when the underlying documents grant access to required structures. Therefore, a document's logical structure needs to be analyzed and recovered. The resulting information pertaining the document structures, e.g. sections, subsections, tables, figures etc. can then be exploited to feed semantic information systems with corresponding ontological information, cf. Document Components Ontology [46, 78], and Core Documentation Entities [64].

In the context of technical documentation, Logical Document Structure Recovery and Analysis is a topic when the source data of documents is either not available or exists in a highly unstructured or proprietary format. Logical Document Structure Recovery and Analysis, in this case, subsequently follows the 1-STAR semantification step described in Section 4.3 and aims on typing and ordering already recovered structures. In literature, Logical Document Structure Recovery and Analysis is sometimes subsumed by the Document Layout Analysis task (see Section 4.3).

This section presents a knowledge-based approach for Logical Document Structure Recovery and Analysis that especially considers the characteristics of technical documents. The remainder of this section is structured as follows: Section 4.4.3 presents the methodology of a novel knowledge-based Logical Document Structure Recovery and Analysis approach. Section 4.4.4 presents a tool for the targeted and interactive acquisition of the required formal knowledge. Section 4.4.5 reports on some practical

aspects regarding the knowledge acquisition in real-world projects. In Section 4.4.6 resulting electronic formats are discussed. Section 4.4.7 gives practical recommendations for applying Logical Document Structure Recovery to technical documents. Section 4.4.8 presents related work.

### 4.4.3 Logical Document Structure Recovery and Analysis

Most of the work on document structure recovery concentrates on scientific articles. While the basic methodology of (1) identifying contiguous text blocks, (2) classifying these text blocks, and (3) post-processing these classified text blocks is similar, the presented approach introduces novel aspects that especially improve the recovery of structures in technical documents. The problem of document structure recovery is defined as follows:

**Definition 4.4.1** (Logical Document Structure Recovery). Given a document  $D$ , first find contiguous blocks of text  $b$ , such that document  $D = \cup b$ . Then, for each contiguous block of text  $b \in D$  find a logical class  $C$  that correctly represents its semantics.

Similar to scientific articles, technical documentation usually follows special (corporate) style guides. Hence, the basic formatting of technical documentation is assumed to be constant for longer periods of time, i.e., larger portions of a legacy document corpus. In contrast to scholarly works, however, the number of classification targets, i.e., structures that shall be recovered, is much higher. For a regular scientific paper the number of structures is assumed to be around 40 [122]; a typical technical document can consist of more than 400 structures [194]. Moreover, semantically different structures in technical documents are with respect to formatting much more similar among themselves as it is the case in scientific articles. A typical example for this is the differentiation between regular ordered lists and descriptions of repair procedures with numbered steps. Due to the usually bigger size of technical documents, formatting must also be assumed to be prone to errors and therefore inconsistent. Another shortcoming of existing solutions is that they can be hardly applied by domain experts/technical writers, i.e., the people that know best about the logical structure of technical documents. These people are usually unfamiliar with the underlying knowledge representations like business rules or the correct configuration of machine learning algorithms and the definition of the required training data. Thus, the application of existing approaches for structure recovery is in most cases not practical for technical documents. Therefore, the presented approach adapts the idea of exploiting formatting information but operates on a knowledge representation that (a) appears more natural for domain experts and (b) is more flexible with respect to inconsistencies and similarities of and between document structures, see Section 4.4.3.

Figure 4.7 gives an overview of the complete process that is divided in a training phase and a recovery phase. The training phase starts with a sampling step that chooses documents from the corpus that are used during the training phase. A micro structure detection is performed on the sampled documents to identify blocks, cf. Section 4.3. Then, a feature selection step generates classifier information, i.e., an alignment of block features to classification targets (document components described in an ontology). After the training phase, the generated classifier gets applied to the complete corpus. The classification itself is based on a standard problem-solving algorithm and is accompanied by an ontology-based optimization step, see Section 4.4.3.

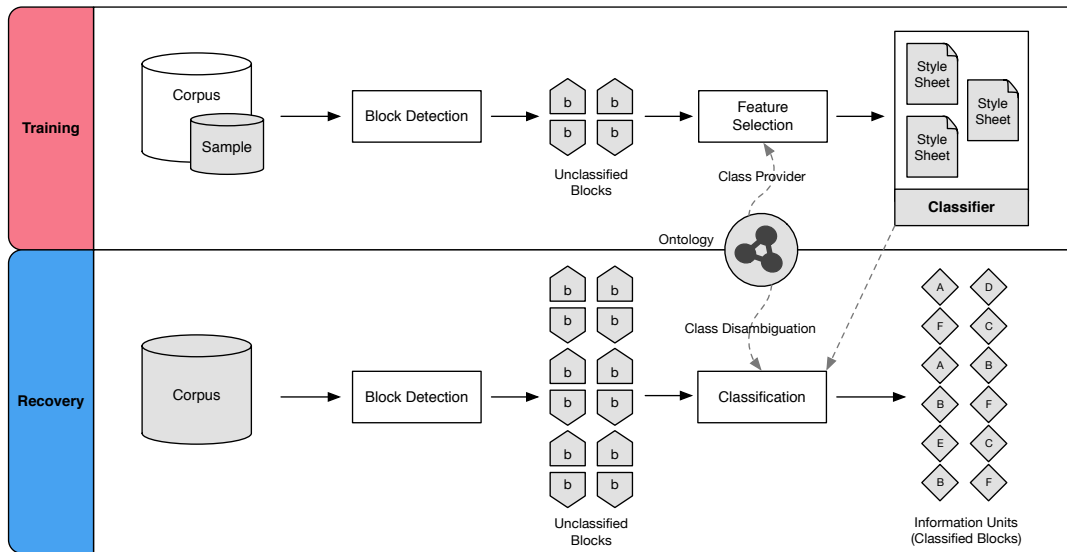


FIGURE 4.7 | Overview of the recovery and classification phases.

### Knowledge Representation

The Logical Document Structure Recovery and Analysis approach presented in this section heavily relies on background knowledge. First, this is a document structure ontology (see Section 4.4.3) that provides the classes that shall be used to type micro structures. This ontology usually also models constraints regarding the usage of classes. Additionally, a knowledge base is employed that consists of set-covering models (see Section 4.4.3) providing the classification knowledge that is required to actually assign type information to specific micro structures.

### Document Structure Ontology

The goal of the Logical Document Structure Recovery and Analysis approach presented in this section is to correctly classify a set of unclassified micro structures. The required classes are externally provided by a document structure ontology. In the context of technical documents, such a document structure ontology usually corresponds to a formal information model for technical documentation, cf. Section 3.4. An respective information model must at least describe structural components that might occur in technical documents and relations that are possible between these structures. Document structures that shall be recognized during the recovery process should be available as classes in the information model. These classes are instantiated

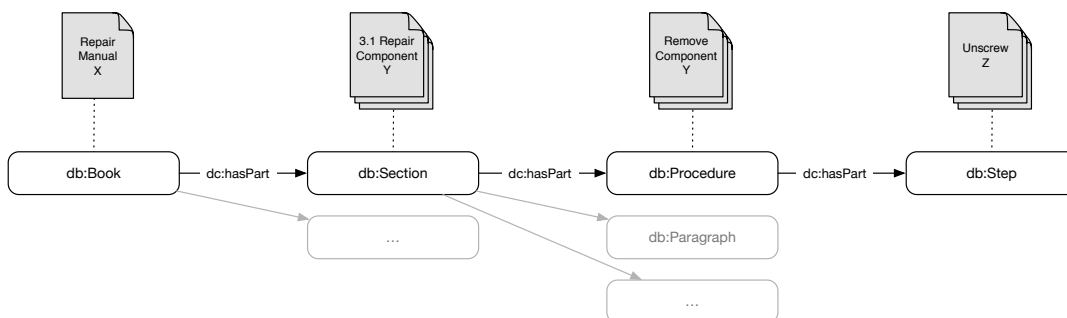


FIGURE 4.8 | Document Structure Ontology according to the DocBook information model.

for concrete text micro structures after the classification process. For the representation of relations between micro structures respective properties must be provided by the ontology. Figure 4.8 shows an exemplary ontology structure with classes and relations derived from the DocBook information model [194]. The example illustrates that a technical document (book) consists of sections that contain structures like paragraphs, procedures etc. that again contain structures like steps.

### Excursus: Set-Covering Models

In the training phase of the presented approach for Logical Document Structure Recovery and Analysis, classification knowledge for micro structures is acquired. In order to formally represent the gathered syntactic knowledge about document structures the presented approach relies on Set-Covering Models [165, 11]. In the following a brief introduction of the key concepts of Set-Covering Models is given, for more details please refer to the the original works.

In contrast to rules, Set-Covering Models allow for an independent modeling and the inherent support of uncertainty. This is especially a benefit when working with (technical) documents that are inconsistent with respect to formatting. While the usage of rules would require to model all possible inconsistencies, set-covering models can inherently handle such data-side issues.

The key idea of set-covering models is the definition of set-covering relations of the following form:

**Definition 4.4.2.** If a class is assigned, then the parameters (attributes)  $A_1, \dots, A_n$  are usually observed with corresponding values  $v_1, \dots, v_n$ .

A set covering relation is formally defined as  $r = C \rightarrow A : v$  where  $A : v$  denotes a feature and  $C$  a target class. A target class is derived from an ontology describing document structures (see Section 4.4.3). A single set-covering relation expresses that a class covers a feature. Figure 4.9 shows a set-covering model  $R$  with five set-covering relations for the two classes  $C_1$  and  $C_2$ . An edge from a class  $C$  to a feature  $A : v$  with the label  $r$  indicates a set-covering relation  $r = C \rightarrow A : v$ .

The definitions given in the following are required throughout the remainder of this section. We define  $\Omega_C$  as the universal set of all classes. The universe of all possible parameters (attributes) is defined as set  $\Omega_A$  where each parameter  $A$  has a range of values  $dom(A)$ . The universe of all possible values is defined by  $\Omega_V = \cup_{A \in \Omega_A} dom(A)$ . From the universe of all parameters and all values we define the set of all features as  $\Omega_F = \{A : v \mid A \in \Omega_A, v \in dom(A)\}$ . Then  $\Omega_R$  denotes the universal set of all set-covering relations. A set-covering model is then defined as  $R \subseteq \Omega_R$ .

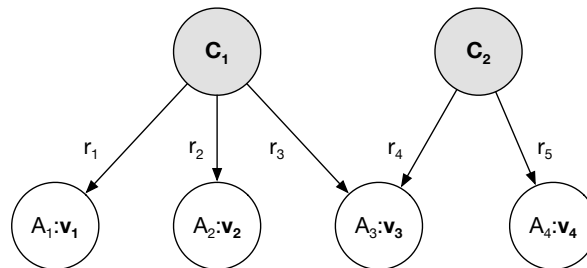


FIGURE 4.9 | Set-Covering model for the classification of the classes  $C_1$  and  $C_2$ , using the parameters  $A_1, A_2, A_3$  and  $A_4$ .

An observation is defined as a subset  $\mathcal{F}_O \subset \Omega_F$  of all possible features. A subset  $\mathcal{H} \subseteq \Omega_C$  of classes is defined as hypothesis. A hypothesis  $\mathcal{H} = \{C_1, \dots, C_n\}$  can be



handled as a conjunction of classes  $C_1 \wedge \dots \wedge C_n$  which shall explain the given observation  $\mathcal{F}_O$ . During the classification the goal is to find the best matching hypothesis  $\mathcal{H}$  that explains all observed features. A hypothesis  $\mathcal{H}$  explains all observed features, if all features are covered by at least one class  $C \in \mathcal{H}$ .

Quality measures are employed for the evaluation of a hypothesis. These quality measures are functions that compute the difference between the observed features  $\mathcal{F}_O$  and expected features  $\mathcal{F}_H = \cup_{C \in \mathcal{H}} \mathcal{F}_C$ , where  $\mathcal{F}_C$  corresponds to the features of class  $C$ . Typical quality functions are *support* and *confidence* that loosely correspond to the well-known concepts of *recall* and *precision* [158]. The confidence expresses how many of the observed features  $\mathcal{F}_O$  are contained in the set of expected features  $\mathcal{F}_H$ . The support in contrast expresses how many of the expected features  $\mathcal{F}_H$  are actually observed.

### Textual Parameters for Set-Covering Models

In the following a specific implementation of set-covering models for the classification of micro structures is described. The universal set of all classes  $\Omega_C$  corresponds to the set of structures that shall be recovered from a legacy document, e.g. section headers on different levels, paragraphs, warnings and notes, tables, ordered and unordered lists, figures, procedures including steps, and many more.

The universes of all possible parameters  $\Omega_A$  and associated values  $\Omega_V$  correspond to features of micro structures and their value ranges respectively. These features can consider different aspects of the structure, e.g. alignment (left, middle, right) and position in page (top, center, and bottom), font and formatting information (font name, bold, italics etc.), statistical features (density, frequencies etc.), page information (page number) and the text itself.

In this context, a set-covering relation is defined as a relation between a micro structure (e.g. section header) and a textual feature (e.g. alignment) including its corresponding value (e.g. left). For a micro structure typically multiple set-covering relations exist. Figure 4.10 shows a (simplified) example of a model that contains set-covering relations for typical headings on level one and level two.

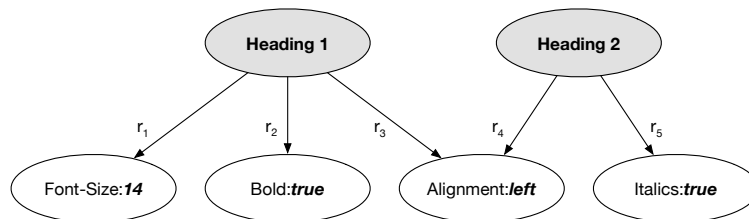


FIGURE 4.10 | Set-covering model for the classification of the classes *Heading 1* and *Heading 2*, using textual parameters.

Listing 4.6 is an example of the set-covering relations in KnowWE [10] markup. The grouped set-covering relations for a single document structure look similar to the well-known Cascading Style Sheets [119] that are used for styling HTML-based web pages. Hence, in the following these relations are referred to as *Classifying Style Sheets*. While the fundamental idea of declaratively describing the style of a piece of text is identical, the usage is reversed. Cascading Style Sheets get applied to elements that have been marked with the corresponding class name, i.e., the class name is matched. In contrast, Classifying Style Sheets require their set-covering relations to be covered from a micro structure.

```

%%CoveringList
Heading 1 {
  Alignment = left ,
  Bold = true ,
  Font-Size = 14
}
@minSupport: 0.5
%

%%CoveringList
Heading 2 {
  Alignment = left [1.0] ,
  Italics = true [1.0]
}
@minSupport: 0.5
%
```

LISTING 4.6 | Set Covering models for Heading 1 and Heading 2.

## Classification

### Set-Covering-based Classification

Once the Classifying Style Sheets have been created, the subsequent classification of complete documents using set-covering models is rather simple: Given a set of observed features, i.e., parameter-value pairs, a hypothesize-and-test strategy [174] is employed. A hypothesize-and-test strategy first picks a hypothesis  $\mathcal{H}$  (i.e. set of classes) and then tests the expected features of the hypothesis against the observed features. A quality measure expresses the covering degree of the hypothesis regarding the observed features in the testing step. Hypotheses are generated and evaluated iteratively until a satisfying hypothesis has been found or all hypotheses have been tested.

Accordingly, text blocks representing micro structures are classified as follows: First the textual parameters and values (observed features) are extracted from an unclassified text block. Then, a hypothesis is generated, i.e., a set of possibly matching classes. Given a set-covering model the covering degree between the observed features from the text block and the hypothesis is computed. Hypotheses with a covering degree exceeding a certain threshold are assigned as possible classes to the text block, i.e., in this phase a text block might have multiple classes assigned. This ensures a high recall.

In the following we formally define the classification of micro structures using set-covering models. A single class assignment for an individual micro structure is defined as triple  $t = \{b, \mathcal{H}, q\}$  with the micro structure in focus  $b$ , the corresponding hypothesis  $\mathcal{H}$  and a quality score  $q$ . For each individual micro structure multiple class assignments can exist, they are grouped in a set  $g_b = \{t_1, \dots, t_n\}$ . The grouped class assignments for all micro structures define the classification result  $\Gamma = \{g_{b_1}, \dots, g_{b_m}\}$ .

### Class Disambiguation

The previous section described the basic classification step for micro structures using set-covering models. The resulting classification  $\Gamma$  is assumed to have a high recall but might contain multiple class assignments for single micro structures, due to the uncertainty considerations that were integrated in the underlying set-covering model. The goal of the subsequent classification step is the disambiguation of these assignments,

i.e., improving the precision by choosing exactly one class for each structure. However, the computed confidence values that express how well the corresponding classes match certain micro structures are not sufficient. Choosing the class with the highest confidence value for each text block might lead to an overall classification result with local optima and probable violations regarding document structure constraints (see Section 4.4.3).

Considering the ontology describing relations and constraints between different document structures (see Section 4.4.3) the intermediate classification result  $\Gamma$  is post-processed in order to disambiguate class assignments. A globally optimized classification result  $\Gamma^*$  gets computed. This is a classification result  $\Gamma'$  with a single class assignment  $t$  for each micro structure  $b$ . The globally optimized classification result  $\Gamma^*$  is computed by solving the following equation:

$$\Gamma^* = \underset{\Gamma'}{\operatorname{argmax}} \sum_{g_b} [ \phi(t) + \theta(\Gamma') ]$$

where  $\phi(\cdot)$  denotes a local optimization function for a class assignment of a text block and the function  $\theta(\cdot)$  ensures that the document structure constraints are not violated for a classification result  $\Gamma'$ . The functions  $\phi(\cdot)$  and  $\theta(\cdot)$  can be freely defined. Base line implementations might return the classification confidence for function  $\phi(\cdot)$  and compute a bonus or penalty on the basis of the information contained in the document structure ontology for function  $\theta(\cdot)$ .

### Reading Order Determination

The previous sections described the classification of micro structures. Applying the described approach to a technical document yields a sequence of typed micro structures. However, the correct reading order remains unclear. This work concentrates on the semantification of technical documents for the subsequent use in semantic information systems. The consuming information systems usually exploit the added semantics at retrieval time but present the original documents (e.g. scanned images or documents in PDF format) to the user. Thus, reading order determination is considered a negligible aspect in this work.

Nevertheless, the type information added to micro structures can be exploited to support the determination of the originally intended reading order. Therefore, the underlying document model must define reading order constraints for certain types, e.g., a “step” structure must either follow a structure representing a “Procedure heading” or another “step”. These constraints can then be used together with geometrical information to pair-wise determine reading orders among micro structures. Finally, the pair-wise reading orderings can be extended to a total order by employing topological sorting algorithms.

Reading order determination using topological sorting algorithms has been described multiple times in literature. For example, Breuel [29, 27] described the reading order determination as follows:

1. Find tall whitespace rectangles and evaluate them as candidates for gutters, column separators, etc.
2. Find text lines that respect the columnar structure of the document.
3. Identify vertical layout structure (titles, headings, paragraphs) based on the relationship (indentation, size, spacing, etc.) and content (font-size and style etc.) of adjacent text lines

4. Determine reading order using both geometric and linguistic information.

With respect to the 5-STAR semantification approach for technical documents described in this work, steps (1) and (2) are covered by the 1-STAR semantification. Step (3) could be supported by the type information recovered by the 2-STAR micro structure classification described in the previous sections.

#### 4.4.4 Interactive Knowledge Acquisition

Although Classifying Style Sheets are assumed to have a catchy knowledge representation that appears more natural to humans than rules, their creation remains a cumbersome task when operating without tool support. Especially the proprietary and difficult to access PDF format makes it hard to extract the information about possible parameters  $\Omega_A$  and corresponding values  $\Omega_V$  from a document. Therefore, the visual knowledge acquisition tool “TEKNO Studio” has been proposed, that supports the interactive development of Classifying Style Sheets.

TEKNO Studio (see Figure 4.11) consists of three fundamental views. A sidebar (4.11-1) shows thumbnails of all documents within the active corpus. A click on a thumbnail opens the selected document for knowledge acquisition activities in the main view (4.11-2). The main view allows for the navigation in the document and renders the respective pages. Additionally, for each page the detected micro structures are indicated with yellow rectangles. The model view (4.11-3) allows for the creation, modification, and deletion of Classifying Style Sheets and associated set-covering relations. Therefore, the tool supports the user in assigning parameters and values from manually selected micro structures (possibly from different pages or documents) in the main view to Classifying Style Sheets in the model view. Target classes of Classifying Style Sheets shall correspond to the classes of the underlying document structure ontology. That way, the tool allows the user to interactively choose one of these classes that get loaded automatically from the underlying ontology.

Micro structures selected by the user contain possibly different parameters or parameter values. The condensing of this information into a common Classifying Style Sheet requires an appropriate strategy. In the following this strategy is referred to as function  $\lambda = \mathcal{F}_O \rightarrow R$  that transforms a set of observed features  $\mathcal{F}_O$  to a set-covering model  $R$ .

Trivial implementations of the function  $\lambda$  might simply consider the union  $R_O^+$  or intersection  $R_O^-$  of all features derived from the selected micro structures. However, choosing the intersection of all features usually results in a less specific set-covering model that might introduce unintended inaccuracy as possibly other (semantically different) micro structures also match the narrowed set of features. Choosing the union in contrast makes the model more rigid, because an observed micro structure has to fulfill more expected features. The optimal model is somewhere between the intersecting and the union sets. Thus, a more sophisticated implementation of the function  $\lambda$  has been proposed that considers the quality functions introduced in Section 4.4.3. For the condensing of the micro structures features in a common Classifying Style Sheet a minimum support is defined that the resulting model has to fulfill with respect to the selected text blocks. Then the function  $\lambda$  computes a model  $R_O^- \subseteq R \subseteq R_O^+$  with

$$R = \{R_i | \exists j \text{ confidence}(R_j) \geq \text{confidence}(R_i) \wedge \text{support}(R_i) \geq \text{minSupport}\}$$

where  $R_i$  corresponds to a concrete model where the features observed from the selected micro structures have been condensed.

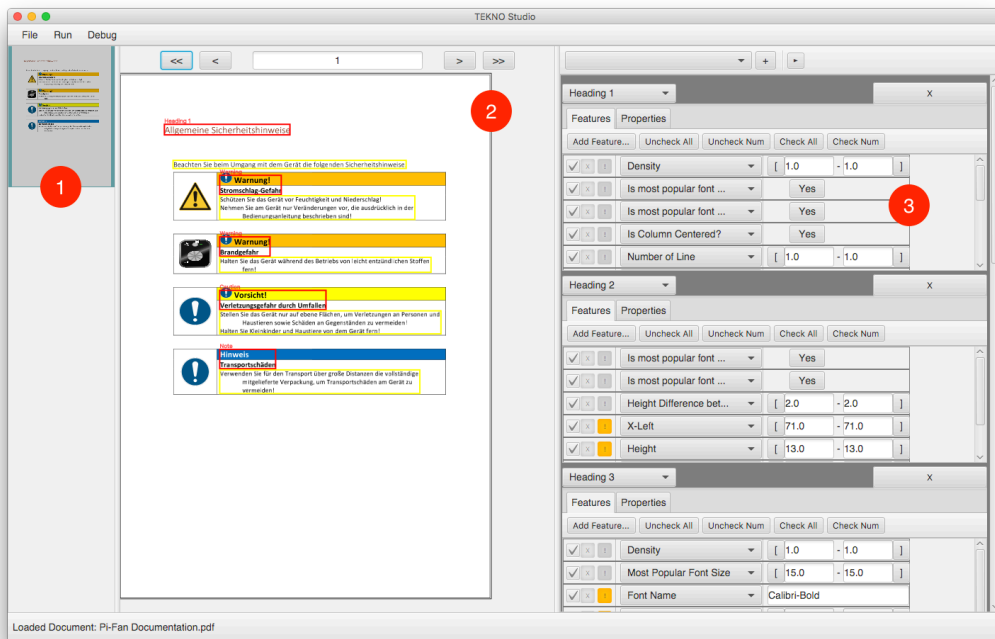


FIGURE 4.11 | TEKNO Studio: Tool support for interactive knowledge acquisition.

Please note that the function  $\lambda$  introduces a user-controlled level of uncertainty. The reason for intentionally introducing and allowing this kind of uncertainty is that in practice corporate style guides are not always obeyed during the authoring of technical documents, i.e., micro structures with the same semantics do not necessarily have the exact same formatting, e.g. slightly varying font-faces or colors.

#### 4.4.5 Knowledge Acquisition in Practice

The interactive knowledge acquisition tool “TEKNO Studio” that was described in the previous section supports the creation of classification knowledge for micro structures. This section states some additional aspects that are relevant for acquiring classification knowledge in real-world projects.

TEKNO Studio provides the following predefined classes that are considered applicable to the majority of technical documents: Heading 1, Heading 2, Heading 3, Heading 4, Title, Table of Contents, Warning, Note, Caution, Caption, Figure, Text, Procedure, and Step. This set of classes can be extended by the user to meet project-specific requirements. Alternatively, the required classes can be read directly from an ontology containing a document model vocabulary.

Although TEKNO Studio provides a set of classes, there is no corresponding predefined classification knowledge. As stated in the previous sections, the respective classification knowledge mainly considers formatting and layout information. Experiences gained in real-world projects show that the respective information is very heterogeneous across corpora of technical documents. Even within the corpus of a single company usually different epochs of documents exist (cf. Section 3.2). Thus, the predefinition of classification knowledge is in most cases not reasonable. Instead, the respective classification knowledge needs to be acquired for *each document corpus* using the described functionality of TEKNO Studio. Therefore, a document corpus

should be partitioned, such that each resulting partition represents a collection of documents that share particular layouts and formatting styles. Then, classification knowledge needs to be defined for each partition using representative samples. The acquired classification knowledge can then be used to classify all micro blocks of documents in the corpus.

#### 4.4.6 Recommended Formats

In general, the formats recommended for 1-STAR technical documents also apply to 2-STAR data. The only requirement regarding the data format is that a type can be explicitly defined. This requirement is especially fulfilled on the presented formats ALTO (practically limited to composed blocks), METS (through the embedded usage of Dublin Core's `dc:type` element), lapft-text (`type` attribute), hOCR (`class` attribute), and PageXML (`type` attribute).

Another, more recommended way is to represent the recovered data according to an information model. The 2-STAR logical document layout analysis is in huge parts a specialized classification task for micro structures. The micro structure classification works upon and exploits class information from an information model. Hence, at the end of the 2-STAR semantification step, an ontology can be populated with micro structures that are represented as instances of their assigned class assignments.

#### 4.4.7 Practical Recommendations

Section 4.4.3 explained the theoretical foundations of the 2-STAR semantification step, which aims on classifying micro structures which in turn gives access to elements like headlines, paragraphs, lists or tables. Section 4.4.6 gave some information about electronic formats that support the representation of the recovery results. This section gives practical recommendations for 2-STAR semantification tasks.

1. **Choose representative samples:**

The described classification is based on set-covering models that are assembled using an interactive knowledge acquisition tool. Although set-covering models are a knowledge representation that inherently support uncertainty it is still important to build up the model from representative examples. Hence, the choice of a representative sample of the complete corpus is critical.

2. **Avoid overfitting set-covering relations:**

For each class one or more set-covering models are assembled. These set-covering models consist of single set-covering relations that correspond to textual features of the micro structures considered for creation. In general, it is good practice to not create set-covering models from single micro structure samples. Multiple micro structures representing the same class should be selected from different pages and/or documents.

3. **Consider regular expressions:**

Depending on the underlying data a plethora of features can be considered for the creation of set-covering relations. Most of these features consider formatting. However, manually defining regular expressions often boosts the performance of the set-covering model based classification.

4. **Avoid regression of set-covering models:**

The continuous addition of set-covering models might lead to a decreasing classification performance. In order to avoid the slowly regression of the knowledge

base an automatic testing environment should be defined that ensures a constant performance for already defined classification knowledge. The development of knowledge-based systems with continuous integration has been described by Baumeister et al. [9].

#### 4.4.8 Related Work

Logical document structure recovery and analysis has a long research history with a perceivable focus on recovering structures in scholarly texts. Multiple prior works surveyed the topic, for an example see Mao et al. [127]. Therefore, this section focuses on works that rely on state-of-the-art methods and are closely related to the presented approach. PDFX [45] uses a two step approach in order to identify 18 typical logical elements in scientific articles. The first step aims on constructing a geometrical model of an article's contents. A subsequent step uses discriminative (statistical) features to identify the different logical units in a predefined prioritised manner. LAPDFText [162] supports the automated decomposition and conversion of PDF files of research texts into a simple text format. The approach relies on three consecutive steps, where text blocks are identified, classified, and finally composed for text extraction. The classification step is rule-based [7, 33] and thus adaptable to new styles and formats of scientific articles. SectLabel [122] is another tool for the structure recovery of scientific articles. In contrast to the knowledge-based tools PDFX and LAPDFText, it uses conditional random fields [113] that are trained with hand-crafted sample data. A general purpose tool for rule-based pattern recognition is UIMA Ruta [106]. The tool consists of an analysis engine that interprets and executes a rule-based scripting language and an authoring tool that supports the development of corresponding rules. In the fields of technical documentation, Oevermann [147] claims to reconstruct semantic structures in technical documentation with vector space classification. However, the text segmentation to micro structures seems to be not very sophisticated and rather inaccurate. Additionally, the blocks are not classified according to a structural but to a rhetorical class like maintenance, transport, or safety.

## 4.5 3-STAR

This section describes the third semantification step. From the micro structures that have been recovered in the 1-STAR and 2-STAR semantification steps macro structures are deduced. This enables the decomposition of large technical documents into smaller, self-contained pieces of information, which is an essential step towards accessible and linkable technical documentation.

### 4.5.1 Information Systems Use Case

In recent years mobile information systems realized as thin-clients on different devices became popular. The main difference to classical information systems is that they are designed for mobile usage. This includes the dynamic retrieval of relevant resources from a central server. The transfer of a complete technical document from the central server to a mobile client is usually not reasonable as the file sizes of technical documents are often hundreds of megabytes or even gigabytes and transfer rates are usually limited. In order to provide quick, dynamic, and modular access to relevant resources, technical documents need to be decomposed into smaller, more fine-grained pieces. However, 1-STAR or 2-STAR technical documents are not yet decomposed.

### 4.5.2 Problem Description

The 2-STAR semantification step yields typed micro structures. The goal of the 3-STAR semantification is the subsequent recovery of a document's macro structure. Original technical documents, especially in unstructured formats like PDF, shall be transitioned into self-contained, modular pieces of information. These self-contained and modular pieces of information are referred to as information units in the following.

**Definition 4.5.1** (Information Unit). Let  $K$  be the complete corpus of technical documents,  $D_i \in C$ . Let  $I$  be the universal set of all information units  $D_{i,j} \in I$ . Then a sequence of information units  $D_{i,j} \in D_i$  forms a document, such that a function  $segment : D \rightarrow 2^I$  can split the document  $D_i$  into information units  $D_{i,j}$ .

Additionally, information units can reference equivalent content in different languages. This is especially important for technical documents, as they get usually translated to a plethora of languages. The ability of information units to reference equivalent content in different languages facilitates the monolingual application of subsequent text analytics steps as results can usually be transferred to all languages. This means, when equal modules in different languages are aligned to an information unit it is usually sufficient to perform all analytics tasks like information typing or semantic annotation for only one of the respective languages. In order to yield (multilingual) information units, a series of semantification steps is necessary. Hence, this chapter focuses on the presentation of respective semantification steps:

#### 1. Macro Structure Recovery

Complete documents with already recovered typed micro structures need to be transformed into a macro structure hierarchy. Depending on the source format, this is a major task. While it is relatively easy for most structured XML-based formats, it is a quite challenging task for closed formats like PDF. An approach for splitting technical documents to a macro structure hierarchy is described in Section 4.5.3.



## 2. Deduplication

Especially in technical documents that were originally issued as printed versions duplicate content is a major challenge. While duplicate content might be helpful in printed books it is usually unnecessary and rather disturbing in dynamic technical documentation. Section 4.5.4 describes approaches for detecting duplicate content in technical documents.

## 3. Alignment

The recovery of macro structures must be performed for each language variant of a document. Reasons include that some languages are more expressive than others and thus lead to different amounts of words and pages per macro structure. The macro structures recovered for each language variant of a document subsequently need to be aligned. Section 4.5.5 describes approaches that promise reasonable applicability for technical documents.

### 4.5.3 Macro Structure Recovery

The 2-STAR semantification step yields typed micro structures like headlines, paragraphs, footers or headers. The basic task in the 3-STAR semantification step is the recovery of a documents' macro structures. The Macro Structure Recovery approach presented in this section transforms an original document  $D_i$  into information units  $D_{i,j}$ . Technical documents are often structured hierarchically, e.g., a section describing the replacement of a component typically has subsections for the disassembly step and the assembly step. Hence, a complete document shall not be decomposed sequentially into a series of macro structures but into a complete hierarchy of macro structures on different levels. Such macro structures usually correspond to chapters, sections and subsections in the original documents.

This section describes a Macro Structure Recovery approach that works upon the typed micro structures recovered in the 2-STAR semantification step. The most relevant micro structures for this processing step are those of type *headline*. Headlines mark start, end, and level of higher level macro structures like chapters, sections or subsections in complete documents (see Figure 4.12).

Algorithm 1 shows a Macro Structure Recovery algorithm that takes as input a document with recovered micro structures and determines a macro structure hierarchy. The algorithm works top-down starting with a macro structure that represents the complete document. The core of the algorithm is the recursive function `createMacroStructures`. This function takes a broader macro structure (parent) and a level information as arguments.

The level information represents the deepness of the macro structures to be recovered and is initially set to zero for the root macro structure. Then, it requests all micro structures representing headlines on the current level that are covered by the parent macro structure. For the first recursion this means that the top level headlines of the complete document are used. This usually corresponds to the headlines of the main chapters of the document. Now, given an ordered list of all these micro structures (headlines) the algorithm actually creates macro structures. This is done by first initializing a new macro structure with the basic information of the corresponding headline (start, level). As the beginning of a new macro structures inherently marks the end of the previous structure on this level this information is used to close the previously created structure. The very last structure on the level needs to be closed using the information of the parent macro structure, i.e., the end of the parent structures is assumed to be the end of the last child macro structure. The closing of created macro structures might consider header and footer information

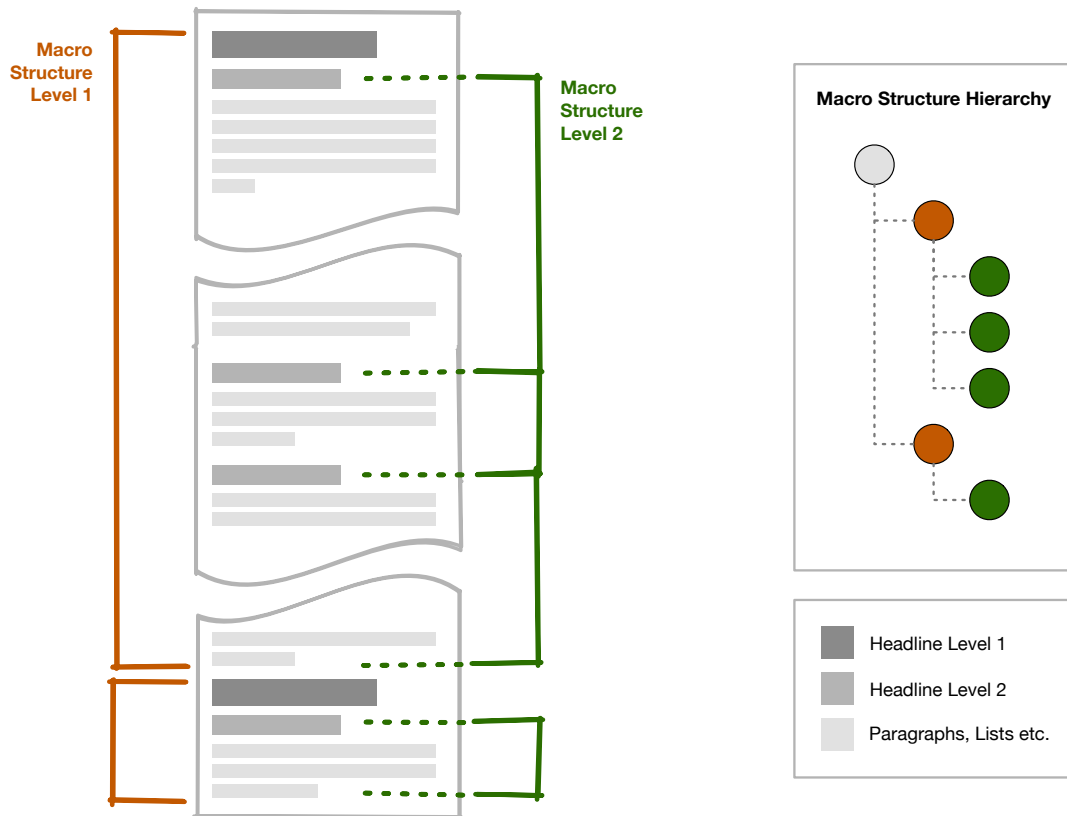


FIGURE 4.12 | Macro Structure Recovery from Micro Structures.

to adjust the actual end of the chapter by excluding them if they appear at the very end of a macro structure. The recursion ends when no more micro structures on the next level exist for a macro structure.

#### 4.5.4 Deduplication

Section 4.5.3 described an algorithm that is able to recover a document's macro structure by exploiting micro structure information. Thus, the basic requirement of the third maturity level to provide access to macro structures have already been fulfilled. However, this maturity level also requires macro structures to be available as deduplicated modules in order to provide a single point of truth for a specific subject. Algorithm 1 is able to perfectly recover a document's macro structure. However, the resulting macro structure hierarchy—depending on the original document—might still contain duplicate modules. Hence, this section presents different approaches to detect duplicate content among the generated macro structures. The main challenge is that the same content might occur redundantly but not identically at different places. Essentially, this is a macro structure *similarity* problem. Considering the characteristics of technical documents and the already available accessibility different approaches can be applied to measure the similarity between macro structures. Section 4.5.4 briefly introduces approaches for measuring the similarity between macro structures' texts. The approach presented in Section 4.5.4 takes the special characteristics of technical documentation into account to determine the similarity between macro structures. Section 4.5.4 presents a similarity approach that exploits types and order of contained micro structures for the similarity measurement.

---

**Algorithm 1** Recursive algorithm for Macro Structure Recovery.

---

```

1: // Start recovery: Create root macro structure for complete document
2: root ← MacroStructure.new
3: root.begin ← document.begin
4: root.end ← document.end
5: root.level ← 0
6: CREATEMACROSTRUCTURES(root, 0)
7:
8: // Recursively create macro structures on different levels
9: function CREATEMACROSTRUCTURES(parent : MacroStructure, level : int)
10:   headlines ← GETCOVEREDMICROSTRUCTURES(parent, HEADLINE, level)
11:   while HASNEXT(headlines) do
12:     headline ← NEXT(headlines)
13:     current ← CREATEMACROSTRUCTURE(headline)
14:     if previous is not null then
15:       previous.end ← current.begin - 1
16:       CREATEMACROSTRUCTURES(previous, level + 1)
17:     end if
18:     if not HASNEXT(headlines) then
19:       current.end ← parent.end
20:       CREATEMACROSTRUCTURES(current, level + 1)
21:     end if
22:     previous ← current
23:   end while
24: end function
25:
26: // Actually create a single structure
27: function CREATEMACROSTRUCTURE(headline : MicroStructure)
28:   structure ← MacroStructure.new
29:   structure.begin ← headline.begin
30:   structure.level ← headline.level
31:   return structure
32: end function

```

---

## Text Similarity

The most basic way to compare macro structures regarding their similarity is the consideration of its texts. Redundant content is usually introduced in different ways:

### 1. Inclusion:

Especially newer technical documents were originally created using an enterprise content management system (CMS). A key feature of such systems usually is content reuse. Hence, recurrent content is often outsourced to respective text building blocks. Typical examples for such contents are environment notices. As the respective text building blocks are normally included automatically at export time this kind of redundancy can usually be detected using simple string matching techniques.

### 2. Copy & Paste:

The most common source of redundant content is copy and paste. In the context of technical documents this especially happens when a basic description of a section appearing early in the document gets detailed in a later section. This is

often the case for technical descriptions and operating instructions. While users are usually satisfied with simple and basic instructions, maintenance personnel require a more in-depth description of components and functions. This kind of redundancy is often accompanied by small changes to the copied text and thus requires a fuzzy comparison.

The text-based similarity of two macro structures can be measured using distance functions. They usually express how many edit operations are necessary to transform one text into another. As edit operations typically the insertion, deletion and substitution of single characters get considered. The most recognized distance function for texts is the Damerau-Levenshtein distance [49]. As the distance is usually expressed as an integer value, different approaches for the conversion to a similarity value exist. A popular approach is to compute a normalized distance, i.e. to divide the distance by the average length of the texts of the considered macro structure  $s_{j,k}$  and  $s_{l,m}$ :

$$\text{similarity}(s_{j,k}, s_{l,m}) = \frac{\text{distance}(s_{j,k}, s_{l,m})}{\text{average}(\text{len}(s_{j,k}), \text{len}(s_{l,m}))}.$$

Determining macro structure similarity based on the corresponding texts usually works well for texts in the same language that provide a decent level of similarity. However, this basic approach can not handle higher amounts of textual differences, which might be introduced by using different technical terms.

### Similarity based on Special Characteristics

The characteristics of technical documents allow for another way of determining the similarity between macro structures, as technical documents contain a lot of information pieces that are almost independent from the surrounding content. These information pieces typically appear on the nano level. Examples comprise numbers accompanied by units (liter, kilograms, tones etc.) and images. Such nano structures can usually easily be isolated by employing pattern recognition techniques [106, 96] that normally work well on text that has been prepared using basic Natural Language Processing techniques like tokenization [132], part-of-speech tagging [32, 164], and shallow parsing [178].

Images usually get transformed to feature vectors. These feature vectors are based on a specific image descriptor. A basic example of an image descriptor is the color histogram method [133], i.e., the color of each pixel of an image is used to fill a histogram. This way, rotations and scalings that might have applied to similar images get ignored during the estimation of the similarity. More elaborated image descriptors determine other image features [37, 196, 123]. Rui et al. [169] surveyed different image descriptors with respect to their usage in image retrieval applications, which mainly requires a good similarity estimation.

Finally, when relevant nano structures have been extracted and a vector representation for images is available the similarity can be determined. However, the computation of the similarity must be separated for the text-based elements and the images. An established way to measure the similarity between two sets of textual elements is to first find a reasonable vector representation. Popular examples for such vector representations comprise bag-of-words, the transformation to TF-IDF [172] values or the usage of word embeddings [137]. When the text-based elements have been transformed to the desired vector representation the respective vectors can be compared using established similarity metrics. Huang [94] surveyed different similarity measures for texts. A popular similarity measure is the cosine similarity:

$$sim_{cos}(s_{j,k}, s_{l,m}) = \frac{vector(s_{j,k}) \cdot vector(s_{l,m})}{|vector(s_{j,k})| \cdot |vector(s_{l,m})|},$$

where the function `vector` transforms the macro structures  $s_{j,k}$  and  $s_{l,m}$  to vector representations. For measuring the similarity based on the contained images a basic approach is to determine the number of similar images in both elements:

$$sim_{img,\alpha}(s_{j,k}, s_{l,m}) = \frac{|\{x|x \in images(s_{j,k}) \wedge y \in images(s_{l,m}) \wedge sim(x,y) \geq \alpha\}|}{|images(s_{j,k})| + |images(s_{l,m})|},$$

where the function `images` extracts the set of images of the macro structures  $s_{j,k}$  and  $s_{l,m}$  respectively in a vector representation. Images can be considered to be similar if the difference between their respective vector representations according to function `sim` does not exceed a certain threshold  $\alpha$ . The overall similarity between the macro structures  $s_{j,k}$  and  $s_{l,m}$  can be determined by computing the harmonic mean:

$$similarity(s_{j,k}, s_{l,m}) = 2 \cdot \frac{sim_{cos}(s_{j,k}, s_{l,m}) \cdot sim_{img,\alpha}(s_{j,k}, s_{l,m})}{sim_{cos}(s_{j,k}, s_{l,m}) + sim_{img,\alpha}(s_{j,k}, s_{l,m})}.$$

### Micro Structure Sequence Similarity

Another way of measuring the similarity between macro structures is to compare the sequences of their contained micro structures. The idea is inspired by the alignment of protein or nucleotide sequences in bioinformatics [77]. There the similarity of sequences is computed in order to identify potential functional, structural, or evolutionary relationships between them. While the goal for technical documents is a different, the underlying idea and the computation is similar:

1. **Create Sequence:**

First of all, a macro structure needs to be transformed into a sequence (vector) of micro structures. This is the fundamental step that allows to employ sequence alignment algorithms.

2. **Create Scoring Function:**

Then, a scoring function must be defined. This scoring function must be able to pairwise determine the similarity of micro structures and to handle gaps in sequences. Therefore, the scoring function should exploit (1) the textual information carried by the respective structures and (2) consider the similarity between micro structure types. The latter can be achieved by predefining a scoring matrix that compares respective types of micro structures. In the field of bioinformatics usually proteins get compared with respect to the PAM [50] and BLOSUM [87] scoring matrices that have been created statistically.

3. **Compute Similarity Matrix:**

Then, a similarity matrix for two sequences of micro structures gets computed. The entries in this matrix express the pairwise similarity between two micro structures according to the scoring function defined in (2). An efficient implementation for filling the similarity matrix is the Smith-Waterman algorithm [183].

4. **Similarity Score:**

The highest value in the matrix expresses the distance between the two sequences. This value must finally be transformed to a similarity score.

The final distance can be computed on the basis of the computed alignment score:

$$\text{similarity}(s_{j,k}, s_{l,m}) = \frac{\text{distance}(s_{j,k}, s_{l,m})}{\text{average}(\text{len}(s_{j,k}), \text{len}(s_{l,m}))},$$

where the function **distance** retrieves the score from the similarity matrix and the function **len** returns the length of the micro structure sequences of the macro structures  $s_{j,k}$  and  $s_{l,m}$ . Figure 4.13 depicts the complete micro structure sequence similarity computation process.

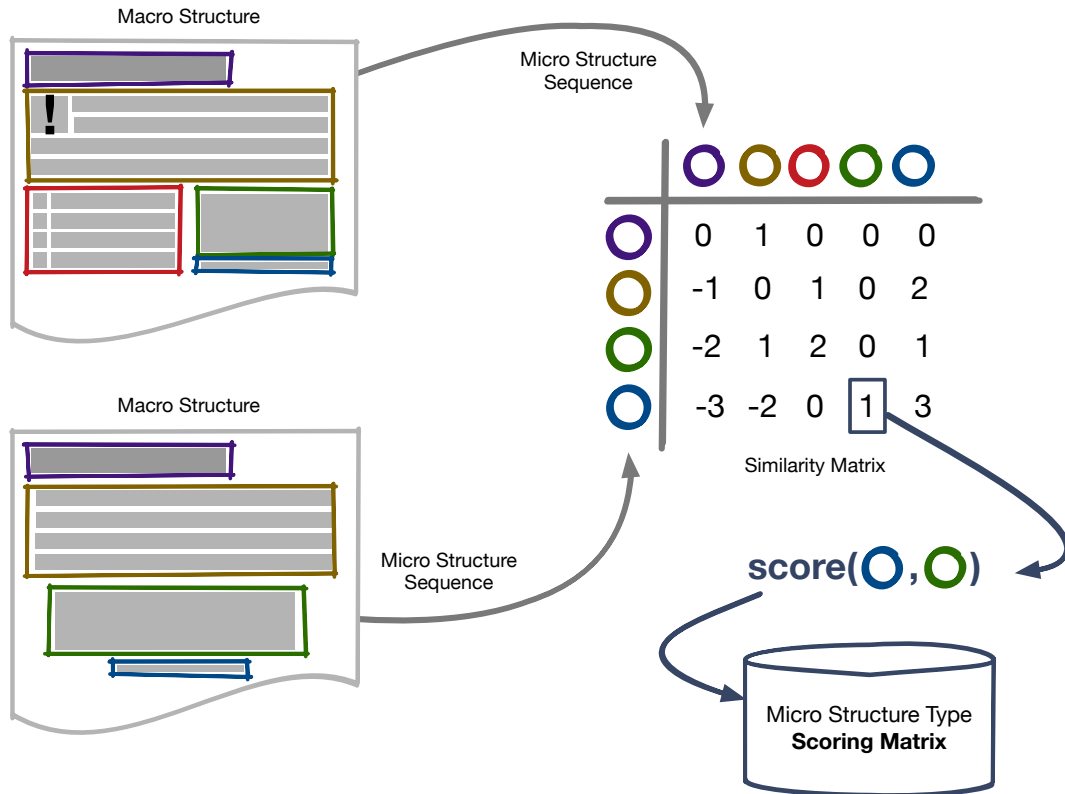


FIGURE 4.13 | Micro Structure Sequence Similarity determination.

#### 4.5.5 Alignment

The macro structure recovery presented in Section 4.5.3 yields a macro structure hierarchy for each processed document. This hierarchy of macro structures undergoes a deduplication process as presented in Section 4.5.4 to eliminate redundant modules. The remaining hierarchies represent potentially equivalent documents in different language versions. In order to achieve the goal of creating multilingual information units as stated as part of the problem definition of the 3-STAR semantification step in Section 4.5.2 potentially equivalent modules need to be aligned. The following sections describe approaches that are able to align hierarchies of macro structures. In case only single modules need to be aligned the methods presented in Section 4.5.4 might also be applied for this purpose. Figure 4.14 illustrates the alignment problem. The illustrated example shows three macro structure hierarchies representing technical documents in language versions for German, English, and Spanish. While the German and English document are structurally equivalent the Spanish document is slightly different.

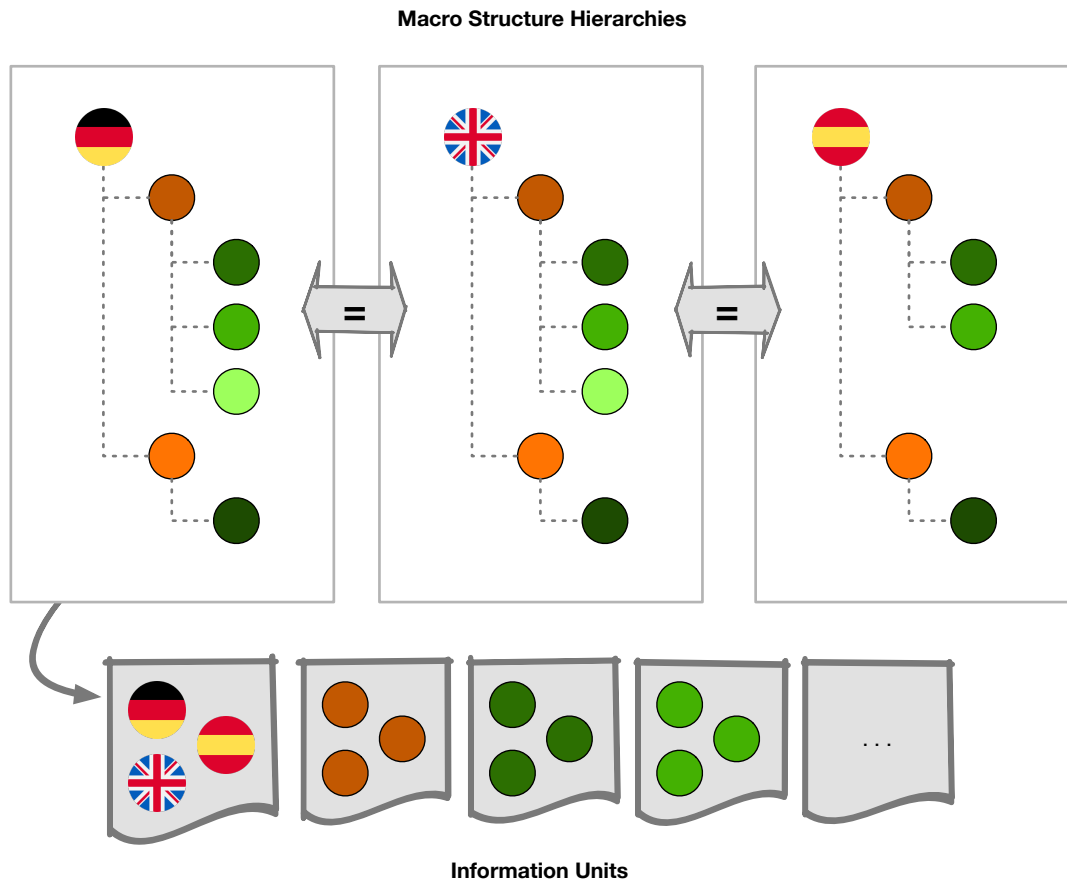


FIGURE 4.14 | Macro Structure Hierarchy Alignment.

### Chapter Number Alignment

Legacy technical documents have usually been published as printed books. Such books usually provide a section numbering schema that is used in conjunction with a section title to identify a specific section. As documents usually get written in one specific source language and get subsequently translated into the required target languages this section numbering often stays constant across language versions. Hence, a naive but in practice successfully applied alignment approach simply aligns modules on the basis of their chapter numbers.

Although this procedure yields good results in practice it is hard to determine the actual correctness of the alignment. In order to get a quantitative indicator of the alignment quality the computation of similarity values according to approaches as presented in Section 4.5.4 and Section 4.5.4 is highly recommended.

### Isomorphic Alignment

In absence of chapter numbering information another way of aligning macro structure hierarchy is to actually exploit the available hierarchy information. As each macro structure hierarchy is essentially a graph (in most cases a tree) a check for isomorphism can be employed to first prove that the respective hierarchies are equivalent and then to align the single macro structures. Unlike general isomorphism checks an actual starting node in the graph of macro structures needs to be selected. This starting node is usually the root macro structure, i.e., the structure that represents the complete document. Then all macro structure graphs get traversed parallelly. During

the traversal for each node the number of ingoing and outgoing edges gets checked which correspond to checking the number of parent and child macro structures respectively. When all graphs has been traversed completely without finding differences in the number of ingoing or outgoing edges of the nodes the macro structures can be considered to be isomorphic. Then another parallel traversal of the graphs can create the actual alignments, i.e. create information unit instances. Algorithm 2 presents the isomorphism check and subsequent alignment in pseudo code.

---

**Algorithm 2** Macro Structure Isomorphism Check and Alignment.
 

---

```

1: // Start check and alignment
2: source ← getHierarchy(sourceRoot)
3: target ← getHierarchy(targetRoot)
4: if ISOMORPHIC(source, target) then
5:   ALIGN(source, target)
6: end if
7:
8: // check for isomorphism
9: function ISOMORPHIC(source : Hierarchy, target : Hierarchy)
10:  sourceIter ← GETTRAVERSER(source, source.root)
11:  targetIter ← GETTRAVERSER(target, target.root)
12:  while HASNEXT(sourceIter) and HASNEXT(targetIter) do
13:    sourceNode ← NEXT(sourceIter)
14:    targetNode ← NEXT(targetIter)
15:    if sourceNode.outDegree != targetNode.outDegree then return false
16:    end if
17:    if sourceNode.inDegree != targetNode.inDegree then
18:      return false
19:    end if
20:  end while
21:  return true
22: end function
23:
24: // Actually create alignments
25: function ALIGN(source : Hierarchy, target : Hierarchy)
26:  sourceIter ← GETTRAVERSER(source, source.root)
27:  targetIter ← GETTRAVERSER(target, target.root)
28:  while HASNEXT(sourceIter) and HASNEXT(targetIter) do
29:    sourceNode ← NEXT(sourceIter)
30:    targetNode ← NEXT(targetIter)
31:    CREATEALIGNMENT(sourceNode, targetNode)
32:  end while
33: end function

```

---

#### 4.5.6 Recommended Formats

The information recovered in the 3-STAR semantification step represents meta information. The actual documents remain unchanged but an additional piece of information gets created that describes the results of this semantification step. The actual result of this semantification step is a set of multilingual information units that refer



to deduplicated macro structure hierarchies. While there are manifold possibilities to represent this kind of information, there are two best practices:

1. **Employ Information Model:**

If the target application is compatible with a standardized information model, the results of this semantification step should be transformed into the respective representation. Such representations are in most cases XML-based.

2. **Standardized and Open:**

If no information model is in active usage, the representation should be as standardized and open as possible. Therefore, the usage of standardized semantic vocabularies should be considered.

Listing 4.7 shows an RDF [107] excerpt in Turtle [15] syntax that represents an information unit that references macro structures recovered from a PDF file in two language variants. The PDF format also supports embedding custom RDF metadata directly into a document. Therefore, Adobe proposed the extensible metadata platform (XMP) [8] that has been standardized as ISO 16684-1:2012. Consumers of PDF documents like semantic information systems can then exploit the XMP/RDF data while indexing resources.

```

:anInfoUnit rdf:type :InformationUnit ;
:hasMimeType "application/pdf" ;
:hasResource
  "sample.pdf?page=3.2-4.7"@en ,
  "beispiel.pdf?page=3.5-5.2"@de ;
:hasTitle
  "Example Macro Structure"@en ,
  "Beispiel Makrostruktur"@de .

```

LISTING 4.7 | 3-STAR data in RDF format.

### 4.5.7 Practical Recommendations

Section 4.5.3 explained the basic recovery process for macro structure hierarchies. Section 4.5.4 presented approaches that aim on eliminating duplicates in these structures. Finally, Section 4.5.5 described how language variants of macro structures can be aligned to form information units. This section gives practical recommendations for the complete 3-STAR semantification tasks.

1. **Sample 2-STAR results:**

The 3-STAR semantification step heavily relies on the results of the 2-STAR semantification. If the micro structure typing is inaccurate the 3-STAR semantification is likely to be even more inaccurate. As a complete check of the 2-STAR results is usually not feasible, the computation of a representative sample is highly recommended.

2. **Exploit PDF bookmarks:**

The 3-STAR semantification mainly exploits micro structures of type headline. As stated before, this is a result of the 2-STAR semantification step and might be inaccurate. Another possibility to identify headlines is to exploit the PDF bookmarks. A lot of PDF documents provide their complete structure in forms of rather accessible bookmarks.

3. **Heuristically group language variants:**

Without background knowledge the described alignment approach needs to

check all documents in the corpus for isomorphism. This is computationally intensive and error-prone. Hence, it is highly recommended to pre-group the corpus with respect to language variants. Usually, heuristics working upon file names or folder structures are a good starting point.

#### 4.5.8 Related Work

In literature, the Macro Structure Recovery is usually referred to as Hierarchical Text Segmentation. Depending on the data quality of the input documents different segmentation methods are used. Established methods are based on lexical or statistical analysis [167, 84, 40, 129, 24, 103].

Soto et al. [185] evaluated different text similarity methods with respect to their applicability to text reuse in technical writings. They considered Cosine Similarity, Longest Commons Subsequence, Google Tri-gram Similarity [99] and Local-Sensitive Hashing [74]. The evaluation is based on four datasets, each representing a book that has been assembled from DITA topics. For their dataset they found the Longest Common Subsequence to be the best performing similarity measure.

The presented alignment of macro structure hierarchies is loosely related to the Parallel Corpora Alignment problem, i.e. the alignment of text blocks or tokens that exist in different languages. This problem has been surveyed by Véronis [193] and Santos [173].

## 4.6 4-STAR

This section describes the fourth semantification step. This step adds information types to the elements of the macro structure hierarchies that have been recovered during the 3-STAR semantification step. This enables the semantic filtering of macro structures (information units) according to an information type, i.e. service technicians are then able to isolate information of a certain type (e.g. repair). This is another remarkable step towards accessible and linkable technical documentation.

### 4.6.1 Information Systems Use Case

The 3-STAR semantification enables information systems to operate on modules instead of complete documents. This yields benefits especially in mobile environments where bandwidth is usually limited. However, as the respective information systems are still based on textual indexes and keyword queries, the filtering of search results for a specific task is still a rather time consuming task. The classification of modules according to information types enables information systems to provide filtering mechanisms. Respective filters usually get realized as faceted search, i.e. a separate multiple choice form that provides check boxes for each class. This way, the search results can be constrained according to rhetorical technical classes like repair, description, or diagnosis. Introducing rhetorical filtering actually introduces basic functionality of semantic information systems to text-based information systems.

### 4.6.2 Problem Description

The 5-STAR maturity schema requires 4-STAR technical documents to be identifiable and information typed. An information type describes the type of the information, e.g. it enables to distinguish descriptive/operational content from maintenance/repair texts. Therefore, the 4-STAR semantification aims on marking macro and micro structures with corresponding information types (see Figure 4.15). These information types need to express the rhetorical aspect of the respective piece of text, see Section 3.3.2. As illustrated in Figure 4.15 information types usually form a taxonomy (for a commonly agreed upon set of information types see Section 3.3.2). The information typing is not limited to but in practice usually applied to macro structures (e.g. chapters or sections) rather than to fine grained micro structures. The addition of information types to already recovered macro and micro structures makes them actual Core Documentation Entities:

**Definition 4.6.1** (Core Documentation Entity). A Core Documentation Entity  $e$  from the universal set of all Core Documentation Entities  $E$  is a self-contained micro structure  $m \in M$  or macro structure  $i \in I$  that is assigned an information type  $t \in T$  that clearly describes its rhetorical aspect.

Although all information typed macro and micro structures fulfill this definition not every combination can be given a label like “repair sequence” or “component overview”. The main benefits of 4-STAR technical documentation is an improved accessibility with respect to rhetorical filtering. This is underlined by the availability of Core Documentation Entities that give access to structures carrying strong technical knowledge.

Additionally, the fourth maturity level requires information typed macro and micro structures to be identifiable, i.e. each information typed structure needs to be assigned an unique and stable identifier. While the uniqueness requirement is easy

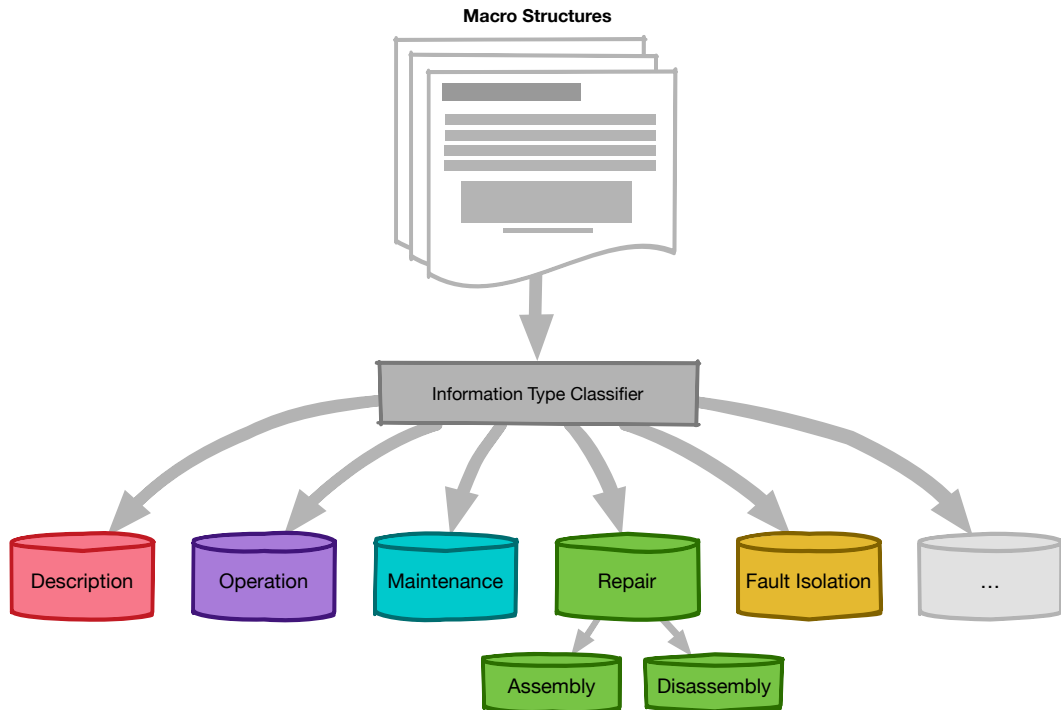


FIGURE 4.15 | Information Type Classification.

to achieve it is usually hard to maintain the identifier across multiple versions of a document that might get processed. The availability of a unique and stable identifier, however, leads to a general improvement of linkability and referability.

Section 4.6.3 maps the information typing of macro and micro structures to the more general Automatic Document Classification problem. Section 4.6.4 discusses the possible absence of training data and gives practical alternatives. Section 4.6.5 gives recommendations for 4-STAR formats and especially points out how to realize unique and stable identifiers.

### 4.6.3 Automatic Document Classification

The information typing of macro and micro structures is essentially equivalent to the Automatic Document Classification problem. Sebastiani [176] surveyed the Automatic Document Classification problem. This section recapitulates the essential statements of Sebastiani's [176] work and complements some state-of-the-art Automatic Document Classification approaches. For a thorough introduction to Automatic Document Classification please refer to the original work of Sebastiani [176].

According to Sebastiani [176] Automatic Document Classification aims on finding a boolean value for pairs  $\langle D_j, c_i \rangle \in K \times C$  where  $K$  corresponds to a set of documents (corpus) and  $C$  to a set of classes. The boolean value *TRUE* for a pair  $\langle D_j, c_i \rangle \in K \times C$  expresses that class  $c_i$  is assigned to document  $D_j$ . The boolean value *FALSE* expresses that for a pair  $\langle D_j, c_i \rangle \in K \times C$  class  $c_i$  is not assigned to document  $D_j$ . There is a general differentiation between assigning the value *TRUE* to exactly one value or multiple values of  $c_i$  per document  $D_j$ . This is referred to as single- or multi-label problem respectively. Information typing macro- and micro structures is a single-label problem, as exactly one information type shall be chosen for a structure. Another differentiation is hard- and ranking categorization. While the aforementioned assignment of boolean values to pairs  $\langle D_j, c_i \rangle$  expresses the hard

categorization, another way of classifying documents is to rank all categories for each  $D_j$  such that the most/least relevant category is top/last.

In order to find the boolean value for a specific pair  $\langle D_j, c_i \rangle \in K \times C$  a classifier is employed. Classifiers can be defined in different ways ranging from heuristic or knowledge-based approaches to machine learning techniques. This section concentrates on supervised machine learning approaches, i.e. classifiers for Automatic Document Classification that get learned from labeled training data. The labeled training data provides documents  $D_j$  with pre-assigned classes  $c_i$ . Supervised Machine Learning techniques exploit the documents' texts and the assigned classes in order to automatically create a classifier.

Machine Learning techniques can usually not operate directly on a document's text. Instead, they need a more compact representation that is in most cases a vector representation. Such a vector representation usually transforms a document's text into a vector of terms with assigned term weights. During this transformation step stop words usually get removed from the text. The terms are then usually represented as a set or bag of words. A popular way to assign weights to the terms is to employ the TF-IDF [172] measure in conjunction with cosine normalization. The term weighting problem for text classification has been comprehensively studied by Lan et al. [114] and Ko [108]. Oevermann et al. [148] claim that using the TF-IDF-CF measure is beneficial for technical documents, as in-class characteristics of tokens get considered.

The resulting vector usually has a high dimensionality. Sebastiani [176] refers to multiple works that state that this is problematic for learning classifiers for Automatic Document Classification. Hence, a commonly agreed upon step is reducing the dimensionality of documents' vector representations. According to Sebastiani [176] this also reduces the danger of overfitting, as observations show that 50-100 training samples per dimension are required to avoid overfitting. As training samples are usually rare, especially in the context of technical documents, fewer dimensions mean less required training data. The dimensionality reduction can be achieved either by building a subset of the existing terms in the vector, which in literature is referred to as *term selection* or by finding synthetic terms. An established approach for the latter is Latent Semantic Indexing (LSI) [91].

Then, having vector representations with a reasonable dimensionality available for documents Machine Learning approaches can be employed in order to find a classifier. A plethora of Machine Learning techniques exists for learning such a classifier. Sebastiani [176] classifies the resulting models to the following categories:

- Probabilistic Classifiers, e.g. based on Naïve Bayes [130, 190]
- Decision Tree Classifiers
- Decision Rule Classifiers [3]
- Regression Methods, e.g. Linear Least Squares Fit [197]
- On-Line Classifiers, e.g. Perceptron, Winnow
- Classifiers using Neural Networks
- Example-based Classifiers, e.g. k-Nearest-Neighbours [82]
- Support Vector Machines (SVM) [124]

Additionally, multiple classifiers can be orchestrated to form a committee [176]. Then, a document gets classified by multiple different classifiers and a strategy is employed

in order to find a common decision for a class assignment. At the time of writing Automatic Document Classification approaches based on Deep Neural Networks also yield promising results [44, 198, 199, 105]. The usage of Deep Neural Networks for Automatic Document Classification usually builds upon word embeddings [137, 138] as vector representation for documents. Goldberg [75] gives a thorough introduction to the usage of Deep Neural Networks for Natural Language Processing Tasks, which comprises the Automatic Document Classification problem.

While the aforementioned approaches rely solely on supervised approaches there also exist unsupervised approaches [128, 181, 135]. These methods usually approach the Document Clustering problem and finally assign classes to the resulting clusters. Unsupervised approaches might be beneficial in industrial use cases where training data for supervised approaches is not necessarily available.

#### 4.6.4 Ground Truth and Practical Alternatives

The previous section summarized existing approaches of Automatic Document Classification for the determination of information types. Automatic Document Classification approaches require a decent amount of training data (Ground Truth) in order to be able to learn a respective classifier. Companies that have already transitioned from classical book-based technical documentation to dynamic module-based documentation (cf. epochs of technical documentation described in Section 3.2) usually produce respective training data when new content is created. This has also been observed by Oevermann et al. [148] who use newly created contents to train a SVM classifier. A classifier that was learned on newly created technical documents can then be employed to add information types to legacy documents. However, not all companies have already transitioned to dynamic documentation that is based on modules. In this case, the following alternatives exist:

- **Create Training Data:**

The most straight forward way is to manually define respective training data. Sebastiani [176] claims that 50-100 examples per class are required to avoid the danger of overfitting. The creation of training data must be evaluated under the cost-benefit-ratio.

- **Train on another Corpus:**

As the information type is considered to be independent from concrete components, functions or other machine-specific entities a classifier might also be trained on a completely different corpus of technical documents.

- **Use Heuristics:**

A practical and in most cases more cost-effective alternative to Automatic Document Layout Analysis is the usage of heuristics that exploit information of the recovered 3-STAR macro structure, i.e., identifying the information type for a complete chapter and then applying it to all successors (sections and subsections).

Such heuristics might exploit the fact that most technical documents are highly structured with respect to their content. For instance, a lot of documents have a certain logical organization, i.e., first describing technical details of components in sections and subsections of chapters named “Technical Description” (or similar), then describing the operation of functions in sections and subsections of a “Usage” (or similar) chapter, and so forth.

If a corpus of technical documents follows such content structures, information

typing can be realized by defining patterns for 3-STAR macro structure hierarchies that are mapped to information types. Such patterns usually specify a target level for elements in the 3-STAR macro structure and a text pattern that needs to be matched.

#### 4.6.5 Recommended Formats

For the representation of 3-STAR semantification results a semantic representation has already been recommended. The main reason for choosing a meta representation was the character of the recovered information. This is also true for the results of the 4-STAR semantification. The determined information type represents additional metadata that gets added to the semantic representation that has been introduced in Section 4.5.6.

Another requirement of the 4-STAR semantification is that identifiers need to be unique and stable. This is a challenging requirement for virtual resources [179], especially when the information is realized as metadata that only references the actual resources. The main challenge is that both the actual resource and the describing metadata might undergo changes that invalidate their resource locations. Possible reasons comprise moving or renaming the actual resource on file servers or additional runs of semantification steps that are not necessarily be aware of an already assigned URL. In literature this problem is referred to as *Digital Curation* [179]. A commonly agreed upon solution to the problem is the usage of persistent uniform resource locators (PURL) [179] instead of standard URLs. PURLs are based on URLs and are thus compatible with semantic technologies. PURLs are intended for referring to virtual resources. In contrast to standard URLs that would directly reference the respective resource, PURLs actually redirect to another URL. While a PURL stays constant the target of the redirection might change. As external applications are then always use the PURL of a resource, respective references are unlikely to break which is an important step towards linkable technical documents. Additionally, PURLs usually use standard HTTP status codes [60] to indicate the status of the respective resource, e.g. when it is temporarily (status code 404) or permanently (status code 410) gone. An implementation of the 4-STAR semantification step is responsible for also maintaining PURLs of processed resources. This means, when an already processed document or an updated version of it gets processed, the redirect targets of existing corresponding PURLs need to be updated. Unlike standard URL definitions PURLs must be managed by a respective system. Such a PURL management system creates new and globally unique PURLs and manages the redirect to the target resources. A popular system for the definition of PURLs is [purl.org](https://purl.org)<sup>15</sup>. Listing 4.8 is an extension of Listing 4.7 that shows the addition of metadata for representing the determined information type and the usage of a PURL as resource identifier.

---

<sup>15</sup><https://purl.org>

```
<http://purl.org/company/book/version/chapter/section>
  rdf:type :InformationUnit ;
  :hasMimeType "application/pdf" ;
  :hasResource
    "sample.pdf?page=3.2-4.7"@en ,
    "beispiel.pdf?page=3.5-5.2"@de ;
  :hasTitle
    "Example Macro Structure"@en ,
    "Beispiel Makrostruktur"@de ;
  :hasInfoType :Repair .
```

LISTING 4.8 | 4-STAR data in RDF format.

#### 4.6.6 Practical Recommendations and Related Work

In practice the key to well-performing automatic document classification is tuning and optimizing the underlying model. In case of technical documentation this especially aims at exploiting the special characteristics of this kind of resource. Oevermann et al. [149] propose several adaptations of established document classification approaches with respect to technical documents. The adaptations consider the special characteristics of technical documents, like standardized patterns, specific terminology, the size of text segments, the availability of training data from Content Management Systems, and quality assurance. The adaptations comprise feature selection, token weighting, term frequency adjustment and confidence scoring. Oevermann et al. [148] also evaluated their adaptations using a SVM classifier. They showed that the technical documentation specific adaptations significantly improved the classification performance on four industrial and thus unpublished datasets.



## 4.7 5-STAR<sup>16</sup>

The 1-STAR, 2-STAR and 3-STAR semantification steps aimed on recovering structural representations of technical documents. The 4-STAR semantification steps added rhetorical information in forms of information types to these structural components. This yields 4-STAR technical documents. In order to receive the fifth star technical documents need to have metadata from an ontology attached.

### 4.7.1 Information Systems Use Case

The results of the 4-STAR semantification enable rhetorical filtering of documents. Technical documents fulfilling the fourth maturity level are usually still used in textual information systems. The availability of information types allows for filtering search results according to a set of information types, which is an important step towards a semantic information system. However, a targeted problem-oriented search is not yet possible as relevant subjects still get queried in forms of textual keywords. Although modern textual information systems can augment keyword-based queries with synonyms the search is still prone to errors due to ambiguities in natural language or inconsistent usage of technical terms. The semantic annotation of structures with concepts from an ontology is the basis for enabling searching with concepts instead of textual keywords. The semantic of concepts is clearly defined and unambiguous due to the availability of additional semantic information, like relations to other concepts, alternative labels or types. In semantic information systems a semantic autocompletion [95] helps the user to define queries by proposing concepts from an ontology on basis of its textual entries and additional context information (search history, already entered concepts, etc.). A semantic search engine then expands the entered query to determine a set of potentially relevant concepts. Finally, the search results get determined by looking up the documents that got annotated with the respective concepts.

### 4.7.2 Problem Description

The 5-STAR semantification described in this chapter aims at the identification and annotation of the main subjects for a given technical document. This is related to the more general problem of *Subject Analysis* or *Subject Indexing*. The approaches for Subject Analysis in technical documents assume that the identifiable subjects can be derived from a domain ontology and rely on a well-defined terminology, providing terms for these subjects. The approaches additionally require that technical documents have been hierarchically structured, i.e. from complete books to macro structures within documents and finally to micro and nano structures representing tokens and terms. Figure 4.16 depicts this hierarchical structure with an example. This brief overview of the problem definition gets detailed in the following subsections by

---

<sup>16</sup>The contents of this section are a slightly extended and reorganized combination of the work described in the following already published articles:

Sebastian Furth, and Joachim Baumeister. "Towards the Semantification of Technical Documents". FGIR'13: Proceedings of German Workshop of Information Retrieval, 2013 [67].

Sebastian Furth, and Joachim Baumeister. "Semantification of Large Corpora of Technical Documentation." Enterprise Big Data Engineering, Analytics, and Management, IGI Global, 2016 [65].

Sebastian Furth, Volker Belli, and Joachim Baumeister. "The Revieal of Subject Analysis: A Knowledge-based Approach facilitating Semantic Search". Proceedings of the Conference "Lernen, Wissen, Daten, Analysen", 2016 [68].

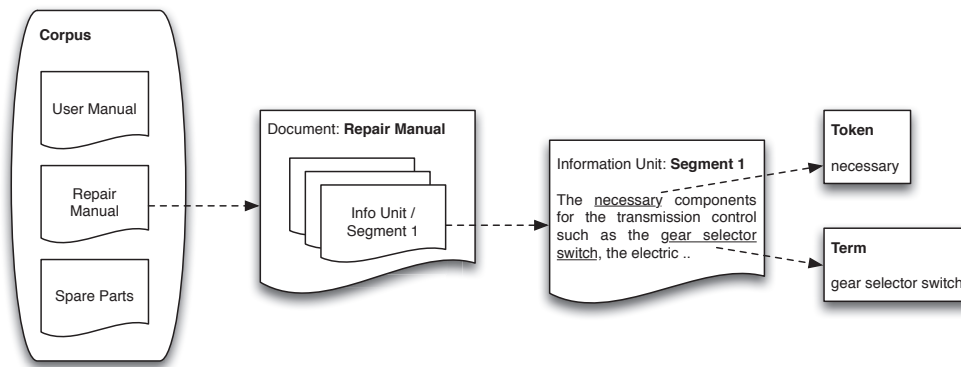


FIGURE 4.16 | The elements of technical documentation.

introducing the set of identifiable subjects, their relations to controlled vocabularies, and the subject indexing problem.

### Controlled Vocabularies

The fundamental requirement for Subject Analysis and the subsequent Subject Indexing is the existence of a controlled vocabulary. Historically, a controlled vocabulary defined the way how concepts were expressed, provided access to preferred terms, and contained a term's relationships to broader, narrower and related terms. Nowadays, such information is typically modeled by standardized ontologies [89, 139, 107], where terms are embedded in complex networks of concepts covering broad fields of the underlying problem domain. Typical examples are ontologies powering semantic enterprise information systems. In such systems users interact using concepts that are known company-wide and valid. An increasing amount of companies maintain corresponding ontologies as they are the key element for the interconnection of enterprise systems and data [188]. If such ontologies do not exist, the construction is usually very reasonable under cost-benefit considerations, as they support not only semantic information systems but are also a vehicle for the introduction of more elaborate services like Semantic Autocompletions [95] or Semantic Assistants. A controlled vocabulary is defined as follows:

**Definition 4.7.1** (Controlled Vocabulary). A controlled vocabulary is an ontology  $O = (T, C, P)$  that contains a set of terms  $T$  that are connected to a set of concepts  $C$ . Concepts  $c \in C$  are connected to other concepts using properties  $p \in P$ .

In the following the existence of a controlled vocabulary for the problem domain in forms of a *domain ontology* is assumed. Considering the technical domain such a domain ontology usually describes the structural decomposition and functions of a machine. Then, the set of identifiable subjects  $S$  can be derived from the domain ontology in forms of ontological concepts.

**Definition 4.7.2** (Identifiable Subjects). Let  $O = (T, C, P)$  be a controlled vocabulary represented as domain ontology, consisting of ontological concepts  $c \in C$ . Then the set of identifiable subjects  $S \subseteq O$  is defined as a set of ontological concepts  $S = \{c_1, \dots, c_n\}$ .

## Entity Recognition

Assuming that such an ontology/controlled vocabulary exists the task is to examine the subject-rich portions of the item being cataloged to identify key words and concepts. Such subject-rich portions correspond to macro structures (information units) that have been created using the 3-STAR and 4-STAR semantification techniques. For each information unit  $i \in I$  occurrences of terms  $t \in T$  from the controlled vocab-

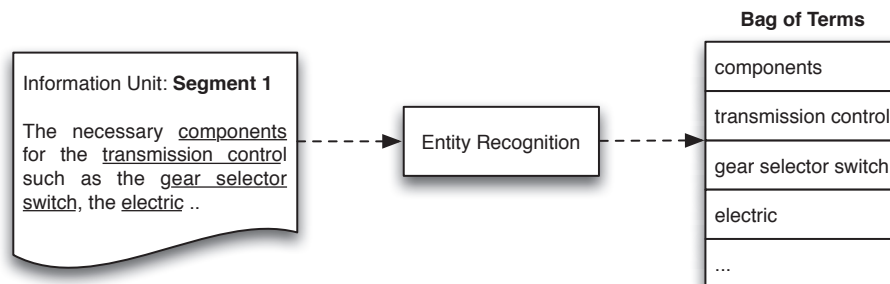


FIGURE 4.17 | Transformation of an information unit to a bag of terms using entity recognition methods.

ulary need to be identified, since the semantic annotation algorithms employed for the subsequent Subject Indexing task operate upon these terms. That way, the domain terminology  $T$  is the basis for an *Entity Recognition* [143] step. An Entity Recognition task is defined as the matching of known entities (terms) from a controlled vocabulary (terminology  $T$ ) in a text. Multiple approaches to Entity Recognition exist. Due to the assumed availability of a controlled vocabulary a dictionary-based entity recognition approach can be employed. The dictionary-based entity recognition method identifies all occurrences of terms  $t \in T$  in available 4-STAR information units  $i$ . The identified occurrences are represented in a bag of terms representation  $e_i$ .

**Definition 4.7.3** (Entity Recognition). Let  $E = \{e_i\}$  the universal set of bags of terms, where each  $e_i = \langle t_1, \dots, t_n \rangle$  is a vector representing a bag of terms recognized in an information unit  $i \in I$  and each  $t_j$  is an element of the domain terminology  $T$ . Then the entity recognition is defined as a function  $er : I \rightarrow E$  that transforms an information unit  $i \in I$  to a bag of terms  $e_i \in E$ .

Practical observations have shown that technical terms are usually not used in a consistent way. Therefore, the dictionary-based matching of terms should use *fuzzy* techniques. Industrially proven fuzzy matching techniques comprise:

- Matching based on word stems that have been produced by a standard Porter stemmer [157].
- Allowing order independent matches for multi-word terms, i.e., consider all (reasonable) token permutations.
- Allowing non-contiguous matches, i.e., ignore non-matching tokens between tokens belonging to a term.

Figure 4.17 shows the entity recognition process for an information unit, that is about the components of the transmission control. The resulting bag of words contains all terms, known with respect to a controlled vocabulary, that have been identified in the information unit, e.g. "components", "transmission control", "gear selector switch", or "electric".

## Subject Indexing

Having information units  $i \in I$  available, the Subject Indexing task can be defined as follows:

**Definition 4.7.4** (Subject Indexing from information units). For each Information Unit  $i \in I$  find a set of concepts  $C_i \subseteq C$  from an ontology  $O$  that describe the topic of the corresponding text best.

For each information unit  $i$  an associated bag of term matches  $e_i$  exists, i.e., a list of terms from a domain ontology/controlled vocabulary that occur in a particular information unit. Given the bag of term matches, the task can be specialized as follows:

**Definition 4.7.5** (Subject Indexing from bags of terms). Given a bag of term matches  $e_i$  determine the underlying topics in the form of a set of concepts  $C_i \subseteq C$  from an Ontology  $O$ .

The availability of formalized domain knowledge is usually a valuable support factor for tasks that cover certain aspects of a problem domain [140, 152, 160]. The following sections show that this is also true for Subject Indexing where the selection of topics can profit from formalized background knowledge. Thus, the integration of domain knowledge in the annotation mechanism (Subject Indexing) becomes a critical success factor and the task can be further refined as follows:

**Definition 4.7.6** (Subject Indexing with background knowledge). Given a bag of term matches  $e_i$  determine the underlying topic in the form of a set of concepts  $C_i \subseteq C$  from an Ontology  $O$  considering the domain knowledge contained in Ontology  $O$  expressed by properties  $P$  between concepts  $C$ .

The following sections present two different approaches to Subject Indexing with background knowledge. Section 4.7.3 introduces Explicit Semantic Analysis that exploits ontological information to build a semantic interpreter. A probabilistic model that exploits ontological background knowledge for Subject Analysis and Subject Indexing is introduced in Section 4.7.4. Section 4.7.5 shows how the results of Subject Analysis and Subject Indexing can be interactively reviewed.

### 4.7.3 Explicit Semantic Analysis

The availability of bag of term representations  $e_i = \langle t_1, \dots, t_n \rangle$  for each information unit  $i$  and an accompanying ontology  $O$  enables the determination of the main subjects of an information unit. Therefore, the recognized terms  $e_i = \langle t_1, \dots, t_n \rangle$  and the background knowledge in the ontology  $O$  get exploited. The goal is to find a set of concepts  $C_i \subseteq C$  from an ontology  $O$  that describe the main subjects of an information unit.

This section describes an approach that has been derived from Explicit Semantic Analysis. Explicit Semantic Analysis has originally been proposed by Gabrilovich et al. [71]. The original Explicit Semantic Analysis was developed for the determination of semantic relatedness between texts and will be briefly introduced in the following. The method is based on a semantic interpreter which copes with a fixed set of concepts, representing each of them as an attribute vector of words. For an experiment Gabrilovich et al. derived concepts from a subset of Wikipedia articles, where each concept corresponded to the title of a Wikipedia article. The words have been extracted from the article text and weights were assigned by using the TFIDF

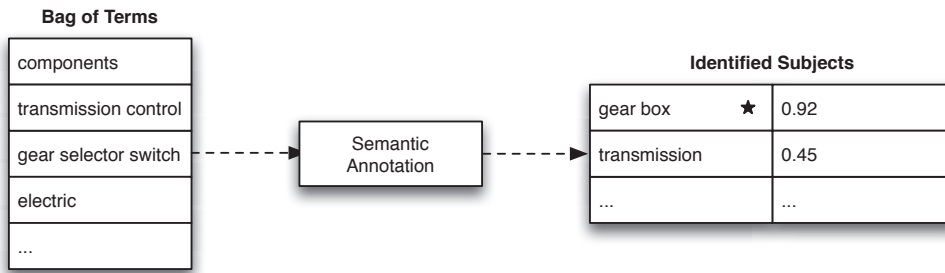


FIGURE 4.18 | Determination of the main subjects on basis of the discovered terminology.

weighting scheme [172]. The semantic interpreter is realized as an inverted index that maps each word into a list of concepts in which it appears. When confronted with an input document, the relevance of the concepts contained in the index can be computed by using the semantic interpreter. For each word in the input document the inverted index is asked for the corresponding concepts and their TFIDF weights. Figure 4.19 shows an excerpt of a semantic interpreter for a gear stick example. Considering that the semantic interpreter is asked for the concepts related to the term  $t_1$  “transmission”, it would return  $o_1$  “transmission” and  $o_2$  “gear box” and their associated weights. The relevance of the concepts is simply computed by aggregating the weights retrieved from the semantic interpreter for all terms  $t_x \in e_{i,j}$ . The result is a weighted vector of concepts, where the top-ranked concept is most relevant for the underlying document (See Figure 4.18 for an example in the gear stick context). The semantic relatedness of texts can then be determined by comparing the computed weighted concept vectors.

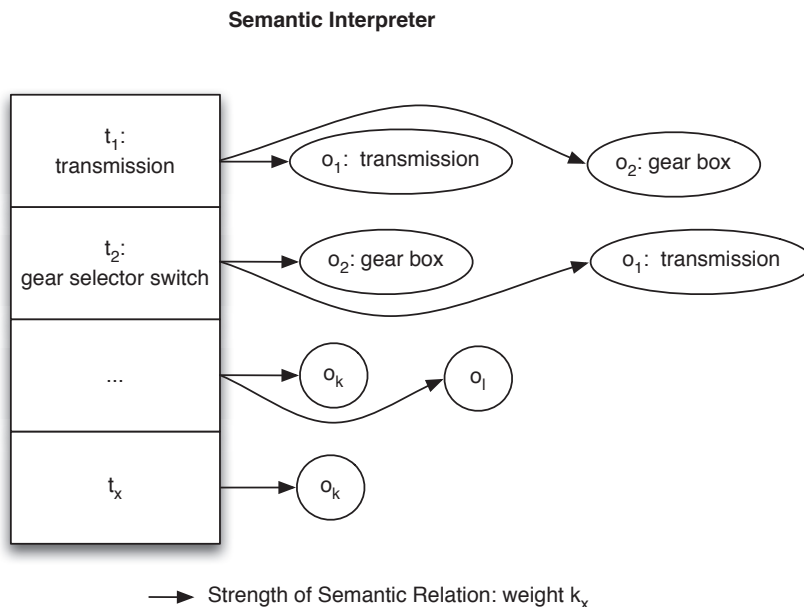


FIGURE 4.19 | An excerpt of a semantic interpreter showing the gear stick example.

### Building the Semantic Interpreter

Similarly to the determination of document relatedness for the Subject Analysis and Subject Indexing task, a semantic interpreter gets employed. However, its purpose is not the determination of semantic relatedness of texts but the identification of the main subjects of an information unit. Instead of TFIDF weights the available domain knowledge in the ontology  $O$  gets exploited to specify the weights, e.g. assuming the set of identifiable subjects  $S$  corresponds to a hierarchy of assemblies, then terms  $t \in T$  for the direct predecessors and successors of an assembly are weighted higher than the transitive ones. Besides hierarchies of assemblies other sources for term-concept relations exist. The availability of a spare parts catalog, for instance, enables the determination of the weight of parts' labels (part of the terminology) as a function of the components they are used in:

- Special parts that are used in only one component of the machine receive the highest weight.
- In contrast, general parts (e.g., bolts) used in many components receive a very low weight.

In the following, let  $\langle k_x \rangle$  be an inverted index entry for term  $t_z \in T$ , where the weight  $k_x$  represents the strength of the association between term  $t_z$  and concept  $s_x \in S$ .

### Using Document Characteristics for Term Weighting

To determine the main subject of a segment, the bag of terms representation  $e_i$  of each information unit  $i$  gets exploited. The bag of terms is the result of the dictionary-based entity recognition method as described in Section 4.7.2. In contrast to [71] the derived approach for Subject Analysis and Subject Indexing also takes document characteristics into account by weighting the terms accordingly. Therefore, micro and nano structures that are available from the 1-STAR, 2-STAR, and 3-STAR semantification get exploited. For instance, the position of a term match in a certain micro structure (headline) might lead to an increased term weight. In the following  $e_i = \{t_z\}$  is the bag of terms representation for the information unit  $i$ , and  $\langle v_z \rangle$  is its corresponding weight vector, where  $v_z$  is the weight of term  $t_z$ .

### Ranking Concepts

The semantic interpreter is then employed to derive a ranked list of concepts for each information unit. Algorithm 3, given as pseudo code, is used to rank the determined concepts. The algorithm basically iterates through all terms  $t_z$  in a bag of terms  $e_i$ , asks for the inverted index entry  $\langle k_x \rangle$  of all concepts  $c_x$  related to term  $t_z$ , and sums up the product of term weight  $v_i$  and relation strength  $k_x$ —this product is referred to as weighted relatedness. The temporary results are saved in a map which gets sorted for the final result in descending order on the weighted relatedness score. This score expresses the relevance of the concepts for the segment, i.e. a higher score means higher relevance.

### Choosing Subjects

The algorithm described in the last section produces a ranking of relevant concepts. Now the most relevant concepts need to be identified—these most relevant concepts are referred to as the *sprint group*<sup>17</sup>.

<sup>17</sup>Corresponding to the sprint group in cycling races that offsets from the peloton.

---

**Algorithm 3** Algorithm for the term-based ranking of concepts.

---

```

1: function GETRANKEDCONCEPTS( $e_{i,j}$ ,  $\langle v_z \rangle$ )
2:    $ranking \leftarrow Map < Concept, Double > .new$ 
3:   for  $t_z$  in  $e_i$  do
4:      $\langle k_x \rangle \leftarrow SemanticInterpreter.get(t_z)$ 
5:     for  $k_x$  do
6:        $wtdrelatedness \leftarrow k_x * v_z$ 
7:        $ranking.update(s_x, wtdrelatedness)$ 
8:     end for
9:   end for
10:   $ranking.sort(WeightedRelatedness, DESC)$ 
11:  return  $ranking$ 
12: end function

```

---

Two different strategies are proposed for the determination of the sprint group. The first one simply uses a threshold, the second one is based on statistical outlier tests. The basic approach for determining the sprint group is taking the score of the most relevant concept. Based on this score all concepts are added to the sprint group that are within a specified threshold, e.g. 90% of the highest score. Basically this yields good results, but there are scenarios where it does not fit. An example for such a scenario is a result where all concepts have low scores, i.e., no concept is actually relevant for the segment. Using the basic approach the majority of the concepts would enter the sprint group. To tackle this issue the usage of statistical outlier tests is proposed. Using such tests it can be determined whether scores exist that offset from the rest. A simple test is for example to compute the interquartile range ( $IQR = Q_{75} - Q_{25}$ ) and then to treat all scores that are higher than  $Q_{75} + \alpha * IQR$  as outliers. In principle, other more sophisticated outlier tests like Grubbs' test for outliers [79] can be applied.

#### 4.7.4 Probabilistic Subject Analysis

Using Explicit Semantic Analysis for Subject Analysis and Subject Indexing already yields good results. A major challenge, however, is comparing the resulting scores. As the scores are not normalized it is hardly possible to compare the quality of the determined subjects between multiple information units. Therefore, this section introduces another approach to Subject Analysis and Subject Indexing that is based on a probabilistic model.

In the following, first a basic probabilistic model that is based mainly on weighted semantic relations between terms and concepts will be introduced. The model can be tailored to integrate expert knowledge for a certain domain specific controlled vocabulary. The basic model will be extended in order to also consider document characteristics, like important document structures (e.g. headlines) or formatting information (e.g. bold text).

##### Basic Probabilistic Model

The basic probabilistic model is founded on observable text/term matches, relations between these terms, and potential topics (concepts) as well as a strong independence assumption between all features. The model connects the features as follows:

1. Starting from a text match  $match \in M_i$  in an information unit  $i \in I$  the model derives potentially corresponding terms  $t \in T$ .
2. The model *optionally* weights the term  $t \in T$  with respect to the covering document structure of the corresponding text match  $match \in M_i$ , e.g., term occurrences in headlines might be more important.
3. Given a term  $t \in T$  the model looks for concepts  $c \in C$  that can be described with this term, i.e. which concepts have this term as label and how specific is this label.
4. The concepts  $c \in C$  derived from the model on the basis of the text/term match  $m \in M_i$  might have relations to topic concepts  $topic \in C_i$  with  $C_i \subseteq C$ . The model exploits ontological information for the derivation of topic concepts  $topic \in C_i$  from observed (term) concepts  $c \in C$  resulting in a topic probability for a text/term match.
5. The derived topic probabilities for each text match  $match \in M_i$  get aggregated in order to compute the overall topic probabilities for an information unit  $i \in I$ .

Given a bag of term matches  $M_i$  for an information unit  $i \in I$ , steps (1) to (3) are realized by computing the topic probabilities for each text/term match  $match \in M_i$ :

$$\mathbf{P}(\mathbf{topic} \mid \mathbf{match}) = \alpha * P(\mathbf{topic} \mid c) * P(c \mid t) * P(t \mid \mathbf{match}).$$

Therefore, the confidence of a term match  $P(t \mid \mathbf{match})$  is considered, i.e. the probability of a certain term  $t$  given a textual match  $\mathbf{match}$ . Additionally the specificity  $P(c \mid t)$  of a term  $t$  for a certain concept  $c$  is taken into account. Unique labels have the maximum specificity of 1.0. The relevance of the concept in focus  $c$  for a topic concept  $\mathbf{topic}$  is  $P(\mathbf{topic} \mid c)$ . This relevance gets computed on basis of ontological information between both concepts. The relevance reaches its maximum if both considered concepts are equal (identity). Finally, the constant prior  $\alpha$  is used to express the linguistic uncertainty that a certain topic is not meant given a certain term match. This avoids that one perfect term match prevents other topics to get more important, i.e. it regulates how many related term matches are necessary in order to outperform one perfectly matching term. Then the topic probabilities for an information unit  $i \in I$  (step 4) get computed on the basis of the topic probabilities for each term match  $match \in M_i$ :

$$\mathbf{P}(\mathbf{topic}) = 1 - \prod_{\substack{M_i \\ match}} (1 - P(\mathbf{topic} \mid \mathbf{match})).$$

The result is a set of topics  $\mathbf{topic} \in C_i$  with associated probabilities that express how well a certain topic fits to the terminology observed in the information unit. This computation assumes independence between the term matches in  $M_i$  according to Bayes' Theorem. The independence assumption might not perfectly reflect reality but is a sufficient approximation in this application scenario.

### Extended Probabilistic Model

The basic probabilistic model can be extended, such that it also considers distinctive document characteristics as valuable background knowledge. In many specialized publications like technical documents or textbooks document structures indicate the



underlying topic or support at least the discrimination of multiple topic candidates. Typical examples are headlines or formatted text (italics, bold, underlined).

The basic probabilistic model uses a constant prior  $\alpha$  that expresses the linguistic uncertainty that a topic is not meant by a certain term match. The basic model gets extended, such that the prior is not constant but depends on the document structure where the term match was observed. Therefore, document structures get weighted according to their importance for the deduction of a topic for an information unit. Assuming that for each document structure a weight  $w$  exists (default 1.0) the value for the prior  $\alpha$  is computed as follows:

$$\alpha_{adaptive} = 1 - (1 - \alpha_{constant})^w.$$

This procedure also allows to discriminate document structures that are inappropriate for the topic deduction, e.g. references/links to other documents.

### Knowledge Representations and Derivation of Probabilities

The preceding sections introduced a simple but powerful and intuitive probabilistic model for Subject Analysis. However, the primary target remains that the Subject Analysis of large document corpora becomes rather a Knowledge Engineering than a Natural Language Processing task. Therefore, the proposed probabilistic model allows for the easy adaptation to characteristics of a domain specific controlled vocabulary and the corresponding corpus of documents that shall be subject indexed. The following sections describe the knowledge-based adaptation, i.e., the definition of basic conditions for the derivation of probabilities.

#### Term Confidence $P(t | match)$

The term confidence  $P(t | match)$  expresses how certain a text match is actually a term occurrence. The computed confidence depends on the quality of the text match. A perfect match, i.e. the text match *match* is equal to the term *t* results in the maximum confidence of 1.0. The usage of fuzzy string matching techniques like order independent matching, stemming etc., might lower the confidence of the term matches. Therefore, implementations of the presented probabilistic approach should allow for the configuration of different fuzziness levels and adjust the confidence accordingly.

#### Term Specificity $P(c | t)$

Given a term, the model must derive all concepts  $c \in O$  that can be described by this term. The model must also express how specific a term is for a concept  $P(c | t)$ , i.e., handle ambiguous terms like “apple” which can be the name of a company or a fruit. In the context of technical documents, terms like “nut”, “engine” or “screw” might be encountered that are very ambiguous and thus unspecific. Therefore, the specificity of a term must be distributed over all potential concepts. In the simplest case the specificity can be distributed equally over all concepts. Unambiguous terms always have a specificity of 1.0. However, experts’ knowledge might be used to prefer certain concepts. This might be useful if some concepts of an ontology are not applicable, e.g., because components they represent are not included in certain machines.

#### Concept Relevance $P(topic | c)$

Then, given a concept, the model must be able to determine how relevant it is for certain topics  $P(topic | c)$ . The procedure is always the same and is explained by the

example of technical documents. In technical documents the occurrence/observation of a concept describing a component might be relevant for a couple of concept topics: (1) machine functions relying on this component, (2) parent components or (3) the component itself.

In general, it is assumed that the relevance of a concept for a topic decreases the larger the distance between both concepts is in the underlying ontology. However, experts' might know that in certain situations (documents) the occurrence of a concept is much more indicative for specific topics than for others. For example in operator manuals, component terms might also indicate functions while they typically do not in repair manuals because usually an operator wants to "operate a function", whereas a technician usually wants to "repair a component".

For the calculation of the concept relevance distances between concepts and topic concepts are extracted/queried from the ontology. Expert knowledge can be used to weight these distances according to the properties  $p \in P$  involved. This way background knowledge regarding the relevance of certain concepts under certain circumstances can be included in the model. Finally the weighted distances between the concept in focus  $c$  and the topic concept *topic* get transformed to a probability. The usage of a normalized sigmoid function is proposed to avoid overestimation of the distance. The parameters  $\beta$  and  $\gamma$  can be used to control the sigmoid function and thus the overall importance of the concept relevance:

$$\mathbf{P}(\text{topic} \mid \mathbf{c}) = \frac{1 + e^{(-\beta)*\gamma}}{1 + e^{(\text{distance}-\beta)*\gamma}}$$

### Linguistic Uncertainty $\alpha$

In the basic probabilistic model the parameter  $\alpha$  is constant. In the extended model the parameter  $\alpha$  can be adjusted, such that it can prefer or discriminate term occurrences in certain document structures. Therefore, domain experts can define weights  $w$  for certain document structures (default 1.0). Values for  $w$  greater than 1.0 prefer, values smaller than 1.0 discriminate terms in certain structures respectively. During the computation of the value for the adaptive linguistic uncertainty  $\alpha_{\text{adaptive}}$  an implementation has to consider the value accordingly.

### 4.7.5 Interactive Review

The Subject Analysis approaches in Section 4.7.3 and Section 4.7.4 yield information units  $i \in I$  that have one or more topics in forms of concepts from an ontology assigned as metadata. In real-world projects the data quality of these metadata is critical. Thus, depending on the requirements on data quality, the manual review of the results of the subject analysis step by domain experts is recommended. The availability of domain experts is usually a crucial element in respective projects (c.f. the well known knowledge acquisition bottleneck). Hence, this section presents an interactive review tool for Subject Analysis and Subject Indexing results (see Figure 4.20). The goal of the presented review tool is to substantially decrease the required review time and thus the availability requirement of a domain expert.

For the review task such a tool needs to fulfill at least the following requirements: (1) Display the hierarchical segmentation of a specific document, (2) display the main subjects for each segment, (3) allow the addition and deletion of subjects. It is additionally proposed to use a visual component that highlights critical annotations in order to optimize the review effort. Critical annotations might be determined on the basis of low confidence values or the semantic similarity of identified subject. The

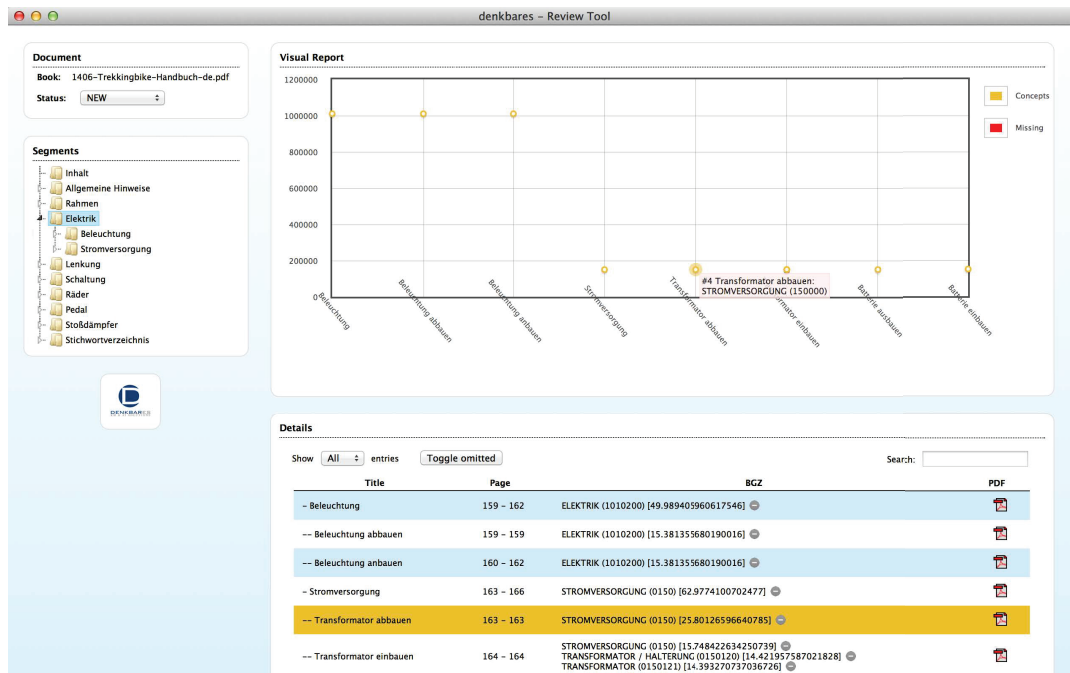


FIGURE 4.20 | A tool for the manual review of semantic annotations, containing the hierarchical segmentation (left), a visual report (top right) and a detail view (bottom right).

latter is especially helpful for technical documents, as observations show that the subject in a sequence of information units is often constant or at least semantically similar. An example for this claim is a technical document that covers the mounting and unmounting of assemblies. In such a document, the probability is high that the corresponding segments of a specific assembly are in a sequence. This results in characteristic patterns that can be visualized using the the visual component. An example is the *semantic staircase* as displayed in Figure 4.20. Additionally, information units without any annotations or with many semantically unrelated annotations should be automatically detected and highlighted.

There exist various metrics for the computation of semantic similarities. Examples for approaches based on WordNet [59] were proposed among others by Jiang et al. [101] or Lin [120]. These metrics might be adapted due to the specificity of the used terminology.

Figure 4.20 shows a sample implementation of a review tool. On the left side the title of the current document is displayed and a status for the document (“new”, “in progress”, “reviewed”) can be specified by the reviewer. Below, the hierarchical segmentation of the document is displayed in a tree view element. The tree view can be used for checking and navigating through the segmentation. Clicking on an element in the tree view loads the information regarding the semantic annotations for the selected segment. The loaded information is displayed in the right part of the application. In the upper part a visual component (“Visual Report”) displays the results based on semantic similarity<sup>18</sup>. Missing annotations are indicated using a red placeholder. At the bottom of the right part detailed information (“Details”) about the semantic annotations are available. They can be accessed by scrolling the view or by clicking on a data point in the visual component. For a thorough review it may be necessary to look up the text of a segment. Therefore the presented review tool

<sup>18</sup>In the example a taxonomy is used for the computation of semantic similarity.

implementation provides direct access to the text in the original document. The detail view also provides possibilities for the addition and removal of concepts supported by semantic typing with autocompletion [95].

#### 4.7.6 Sources of Ontologies

The Subject Analysis and Subject Indexing approaches described in the previous sections require a *Controlled Vocabulary* in forms of an ontology that provide concepts for components, functions, and other entities of the underlying machine (cf. Section 4.7.2). In the past, such ontologies or comparable information often did not exist. Nowadays, an ever-increasing number of enterprises follow the idea of *Linked Enterprise Data* (cf. Section 2.2.4) and, thus, have ontologies readily available. Companies that have not yet started the semantic integration of their enterprise data usually still have multiple data sources that allow the derivation of ontological concepts. The following sections give a brief overview of the most prominent examples of ontology sources in the enterprise context.

#### The Digital Twin

Today, a lot of companies are driven by the ideas of *Digitalization* and *Industry 4.0*. A central part of these ideas is the availability of a so called *Digital Twin*, i.e., data structures that describe components, functions, etc. of respective machines in detail and are valid across multiple departments and processes in enterprises. These digital twins can easily be transformed to an ontological representation. The impact and level of detail of digital twins is underlined by different applications described in literature:

- Rosen et al. [168] discuss the importance of digital twins for the future of manufacturing.
- Boschert et al. [25] discuss the idea of digital twins under simulation aspects.
- Tao et al. [189] report on digital twins within product design, manufacturing and service processes.

#### Master Data / Enterprise Resource Planning

While the idea of digital twins is relatively new, a lot of enterprises maintain *Master Data*. Such master data is usually comparable to the idea of the digital twin, concentrates, however, on information that is required in Enterprise Resource Planning systems. Master data usually describes products that are sold and is often closely coupled to comprehensive bills of materials (BoM) or spare parts lists. Both, original master data and attached materials and parts data can usually be transformed relatively easy into an ontology. Section 7.7 describes the semantification of a corpus of technical documents using an ontology that has been derived from master data. Section 7.4 describes the repeated semantification of corpora of technical documents with concepts derived from spare parts catalogues.

#### Ontology Learning and Population from Text

The previous sections showed that different sources for ontologies exist in enterprise software architectures. However, there are still companies that neither maintain digital twins nor any kind of master data. In such scenarios, ontologies describing machines must be created from scratch. A lot of established Ontology Learning and

Ontology Population [34] approaches exist. These approaches try to learn ontology structures in forms of classes and properties and corresponding instances (population) from text. Although the learning and population of ontologies is not part of the 5-STAR semantification process, the work described in this thesis supports the Ontology Population task. The increased accessibility of semantified technical documents can be used for targeted information extraction that is the fundamental basis for a subsequent Ontology Population step:

- **2-STAR:**

The 2-STAR semantification results in classified micro structures (blocks). Targeted Information Extraction can exploit the textual information contained in certain types of blocks to populate an ontology with instance data. For instance, headlines or image captions might be a valuable source for ontology concepts.

- **3-STAR:**

The 3-STAR semantification yields a hierarchy of macro structures. In technical documents parts of these structures often represent the hierarchical break-down of corresponding machines. Thus, the macro structure hierarchy could be traversed in a targeted manner and considered for populating a technical ontology.

- **4-STAR:**

The 4-STAR semantification adds information types to different types of document structures. As a result, Core Documentation Entities (cf. Section 3.3) like component overviews or procedures are available. These Core Documentation Entities are a valuable sources for the derivation of ontological instance data. Section 7.6 describes in detail the population of large technical ontologies on basis of discovered Core Documentation Entities in the course of a real-world project.

#### 4.7.7 Recommended Formats

The representation of the 5-STAR semantification results builds upon the recommendations for 3-STAR (see Section 4.5.6) and 4-STAR (see Section 4.6.5) representations. The subjects determined using the presented Subject Analysis and Subject Indexing approaches represent additional metadata for already semantically described information units. The Dublin Core [109] ontology provides the `dc:subject` property that can be used for the semantic definition of determined subjects. Listing 4.9 is an extension of Listing 4.8 that shows the addition of metadata for representing the determined subjects (`componentA`). This finally represents the information required for 5-STAR technical documentation. The semantic representation gives access to structural components on a reasonable level, allows to filter structural components according to their information type, and provides subject information that enables a problem-oriented access.

```

<http://purl.org/company/book/version/chapter/section >
  rdf:type :InformationUnit ;
  :hasMimeType "application/pdf" ;
  :hasResource
    "sample.pdf?page=3.2-4.7"@en ,
    "beispiel.pdf?page=3.5-5.2"@de ;
  :hasTitle
    "Example Macro Structure"@en ,
    "Beispiel Makrostruktur"@de ;
  :hasInfoType :Repair ;
  dc:subject :componentA .

```

LISTING 4.9 | 5-STAR data in RDF format.

### 4.7.8 Practical Recommendations

The presented approaches to Subject Analysis and Subject Indexing as well as the presented interactive review tool for semantic annotations have been successfully applied in a series of industrial projects. This sections gives some practical recommendations for the successful application of the presented approaches. The presented approaches are intentionally designed to be rather knowledge-intensive. Hence, the key to a well-performing Subject Analysis and Subject Indexing is the targeted consideration of available domain knowledge. This includes the consideration of available term sources, structured information and relations between single data sets. Additionally, technical writers should be interviewed in order to identify micro structures that should be preferred during the Subject Analysis.

### 4.7.9 Related Work

Topic Analysis is a relatively wide field of research and is strongly influenced by Document Classification and Document Clustering approaches. Notable approaches exist in particular among latent methods, i.e. topics are not expressed in form of explicit concepts but as a set of key terms. Prominent examples are Latent Dirichlet Allocation (LDA)[20] and Latent Semantic Analysis (LSA)[51]. In the presented approach the goal is the identification of a concrete (or explicit) concept, so the latent approaches do not fit for the problem. Efforts have been made to make the topics discovered by the latent approaches more explicit, e.g. Ramage et al. [161] proposed Labeled LDA that constraints LDA by defining one-to-one correspondence between LDAs latent topics and user defined tags. Andrzejewski and Zhu [2] proposed the usage of supervision in form of Topic-in-Set knowledge to improve the recovery of original topics using LDA. Chemudugunta et al. [38] proposed the combination of semantic concepts and unsupervised statistical learning to tag web pages with con-concepts from a known set of concepts without the need for labeled documents. Regarding the deduction of explicit topics, Explicit Semantic Analysis [71] is a well-known approach.

Concerning the review of semantic annotations the authors are not aware of a comparable tool for the review of the 5-STAR semantification results. Ontosophie [36] is a system for the population of event ontologies and uses supervised machine learning for learning extraction rules. These rules also compute a confidence value, which is used to determine whether a human reviewer needs to accept extracted information. The idea of the presented review tool is to guide a human reviewer through an entire book and highlight critical annotations for rapid correction.

## 4.8 Summary and Contributions

Large corpora of technical documents that are not semantically prepared still exist in many companies. This limits their usage in state-of-the-art information systems. This is a severe problem as missing data quickly destroys the acceptance of novel information systems. Hence, there is a fundamental need for migrating legacy documents to formats that are compatible with modern information systems.

This chapter introduced a novel and holistic process that is able to transform documents in legacy formats like PDF to semantic representations. The holistic process is based on five consecutive steps that build upon each other. Existing methods that can be employed to realize the processing steps have been presented. Where necessary and reasonable novel approaches have been introduced. These novel approaches are especially tailored to exploit the characteristics of technical documents as presented in Section 2.1.1.

The 1-STAR semantification (see Section 4.3) presented established approaches from the fields of Document Layout Analysis and discussed different target formats. The 2-STAR semantification (see Section 4.4) introduced a novel micro structure classification approach. This classification approach builds upon the fact that corporate style guides usually fix the appearance of technical documents for a specified period of time. Additionally, the approach takes uncertainty into account, which might occur as formatting can be inaccurate. An interactive knowledge acquisition tool that guides the creation of required classification knowledge has also been proposed for the 2-STAR semantification. The 3-STAR semantification (see Section 4.5) introduced approaches for recovering macro structure hierarchies of technical documents on basis of 2-STAR data. Additionally, similarity metrics have been discussed that support deduplication and aligning documents in multilingual corpora. The similarity metrics discussed especially consider the characteristics of technical documents. The 4-STAR semantification (see Section 4.6) presented Automatic Document Classification approaches which are employed to classify single macro structures (modules) with respect to information types. Finally, novel Subject Analysis and Subject Indexing approaches have been presented for realizing the 5-STAR semantification (see Section 4.7). The presented approaches aim on adding concepts from an ontology as topical metadata to single macro structures (modules). This makes the respective modules usable in state-of-the-art semantic information systems. The approaches are knowledge-based and exploit domain knowledge concerning both the characteristics of technical documents in focus and the underlying machines. An interactive review tool for 5-STAR technical documents has also been proposed, which aims on supporting optional review processes.

The novel and holistic semantification process has already been employed in several industrial projects. The experiences gained have been described in forms of practical recommendations for each semantification step. Additionally, each semantification step presents reasonable formats for the representation of the semantified technical documents.





## Chapter 5

# Implementation of a Semantification Architecture

---

*Knowledge is of no value unless you  
put it into practice.*

*Anton Chekhov*

---

## 5.1 Overview

In this chapter, a 5-STAR semantification tool box is presented. The tool box supports the application of semantification knowledge and methods in an integrated and extensible cloud-based environment called *CAPLAN* (see Section 5.3). The development of required semantification knowledge is aided by an interactive knowledge acquisition tool called *TEKNO Studio* (see Section 5.4). The results of the 5-STAR semantification process can be reviewed using a dedicated *Review Tool* (see Section 5.5).

## 5.2 Distinction: Existing 1-STAR Semantification Tools

Sections 5.3–5.5 describe tools that have been developed within the scope of this work. *CAPLAN* (see Section 5.3) is a cloud-based architecture that supports semantification tasks and provides reference implementations of the described semantification approaches. The focus of *CAPLAN* and its predefined processors is on the 2-STAR, 3-STAR, 4-STAR and 5-STAR semantification steps. *TEKNO Studio* (see Section 5.4) is a tool that supports the knowledge acquisition of 2-STAR classification knowledge. A dedicated review tool (see Section 5.5) guides the (optional) review of 5-STAR technical documentation.

The 1-STAR semantification step is primarily concerned with Document Layout Analysis tasks (cf. Section 4.3.3). Although it is the fundamental basis for the subsequent processing, the Document Layout Analysis task is not specific for technical documents. Additionally, a lot of previous work exists in this field. Therefore, this work considers and recommends the following existing tools and utilities for the 1-STAR semantification step:

- **OCRmyPDF**: For processing scanned images (Optical Character Recognition) this work refers to *OCRmyPDF*<sup>1</sup> that is based on the tesseract library<sup>2</sup>.
- **pdf2xml**: For processing textual PDF documents this work uses the *pdf2xml*<sup>3</sup> utility that converts PDF documents into an XML representation providing access to pages, blocks and tokens.

The tools developed within the course of this work (especially *CAPLAN* and *TEKNO Studio*) are able to import the results of *OCRmyPDF* and *pdf2xml* for the subsequent semantification steps.

---

<sup>1</sup><https://github.com/jbarlow83/OCRmyPDF>

<sup>2</sup><https://github.com/tesseract-ocr/>

<sup>3</sup><https://github.com/kermitt2/pdf2xml>

## 5.3 Using Semantification Methods: CAPLAN<sup>4</sup>

CAPLAN is a cloud-based architecture for the design and application of semantification methods. It is designed to enable a broad field of users to employ the presented semantification approach. Additionally, it is based on an extensible software architecture that allows for the easy extension of additional semantification modules. As semantification projects usually have to cope with large data-sets scalability is an important success factor. Thus, CAPLAN is designed to be runnable in high performance / big data environments.

Many colleagues and students of denkbares GmbH were involved in this implementation project. The development of CAPLAN was supported by the Bundesministerium für Wirtschaft und Energie (BMWi) under the grant ZIM ZF4172701 “APOSTL - Accessible Performant Ontology Supported Text Learning”.

Section 5.3.1 first summarizes the requirements for a cloud-based semantification architecture. In Section 5.3.2, the underlying software architecture of CAPLAN is sketched. Section 5.3.3 actually presents the implementation results.

### 5.3.1 Requirements

The development of CAPLAN began with a formal definition of requirements for a cloud-based semantification architecture. This section briefly summarizes the defined requirements by categorizing them into accessibility with respect to non-expert users, scalability to large-scale data sets and flexibility for new project and semantification requirements. In the following these requirement categories are broken down.

#### Accessibility

The increasing amount of semantification projects requires that semantification processes are accessible for non-experts (with respect to Text Analytics/Natural Language Processing). This requires that the architecture is able to **hide the complexity** of underlying NLP processes. Users without expert knowledge in Natural Language Processing should be able to configure the semantification process on an abstract level without having to know specific details of underlying approaches.

The opening of the semantification process to non-expert users requires that the architecture provides **documentation** for each of the underlying process steps. The documentation for each process step has to state clearly what data in which format is required as input and which results can then be derived from this data as output.

The generated data should be provided with **provenance and versioning information** that states clearly how (which method and parametrization) and when the data has been produced. The availability of such information facilitates the reproducibility of results and the comparison of parameter configurations.

The architecture should also provide ways to **examine generated results** on a high level. Therefore, data visualization techniques should be a vital element in the architecture to open the assessment of the results to a wide user range. Additionally, interactive review tools should allow users to easily correct the generated results.

Another aspect of accessibility affects the representation of the underlying data. Due to their subsequent usage in semantic applications all (intermediate) results

---

<sup>4</sup>The contents of this have already appeared in a preliminary version in the following published article: Sebastian Furth, Volker Belli, Alexander Legler, Albrecht Striffler, and Joachim Baumeister. “CAPLAN: An Accessible, Flexible and Scalable Semantification Architecture”. Proceedings of the Conference "Lernen, Wissen, Daten, Analysen", 2016 [69].

should have a **semantic representation**, i.e. all data elements should at least be identifiable using an URI and provide type information.

### Scalability

Scalability has a two-fold meaning in the context of semantification architectures. It is primarily concerned with the support of **large scale data processing** (Big Data), i.e., the architecture should be prepared to be employed in high performance computing environments for high throughputs. This requires that underlying algorithms are available for Big Data processing frameworks like Apache Spark [134] and the underlying data model supports distributed data storages like Hadoop's HDFS [180].

However, scalability in this context is also concerned with the aspect that a wide range of users should be able to use the semantification architecture. Therefore, the architecture should be realized as **Business Process as a Service**. A business process as a service is typically realized as a cloud service. In the context of a semantification architecture this means that the whole semantification process is available as web application or API.

### Flexibility

A semantification process typically comprises a series of complex operations that successively prepare a resource for the usage in a semantic information system. However, in some cases some of the operations are not necessary, because data is already prepared to a certain extend (cf. 5-STAR maturity schema introduced in Section 3.2). Therefore, users should be able to enter the semantification process at an **arbitrary process step** if they can provide data in the necessary format.

Sometimes the semantification process does not have to be necessarily be completed, e.g., because intermediate results are sufficient for specific application scenarios. Typical examples include specialized Information Extraction tasks that operate on semantically represented micro structures. Hence, the architecture should allow to **query and export intermediate results**.

Although the single steps of semantification processes are usually similar in various application scenarios it might be necessary to parametrize, extend, or adapt the process to new process requirements. Typical scenarios include the existence of a previously unknown source format or new approaches/parameter configurations for specific process steps like segmentation, term matching, or subject indexing. Thus, the architecture shall be **extensible**, such that new process steps or variants of existing process steps can easily be integrated. The extensibility should also be reflected in the data model.

## 5.3.2 Architecture

In the following a software architecture facilitating the semantification of resources under the requirements stated in Section 5.3.1 is presented. Therefore, the key components of the architecture are introduced.

### Components

Referring to the requirements in Section 5.3.1, a semantification architecture should be *accessible*, *scalable*, and highly *flexible*. The flexibility mainly demands for a high extensibility and standardized import, export, and processing functionality in all process steps while accessibility is concerned with hiding complexity from non-experts,

providing easy-to-use assessment and reviewing functionalities and standardized data representations. Thus, the proposed architecture (see Figure 5.1) is composed of interweaved modules, that are represented as quintuples  $Q = \{D, I, P, A, R\}$ , with:

- **Data Nodes  $D$ :** Contain the data and a data description for the process step, e.g., a description of document structures and instance data for concrete documents.
- **Importers  $I$ :** Provide and document import functionalities for data nodes, i.e., describe possible import formats and handle the import of data nodes from raw/source data. Also create provenance information for the imported data.
- **Processors  $P$ :** Process data nodes in order to produce new or update existing data nodes respectively. Also create provenance information for the generated/updated data.
- **Assessments  $A$ :** Provide possibilities/metrics/visualizations to assess a set of data nodes.
- **Reviews  $R$ :** Allow manually changing/reviewing existing data nodes.

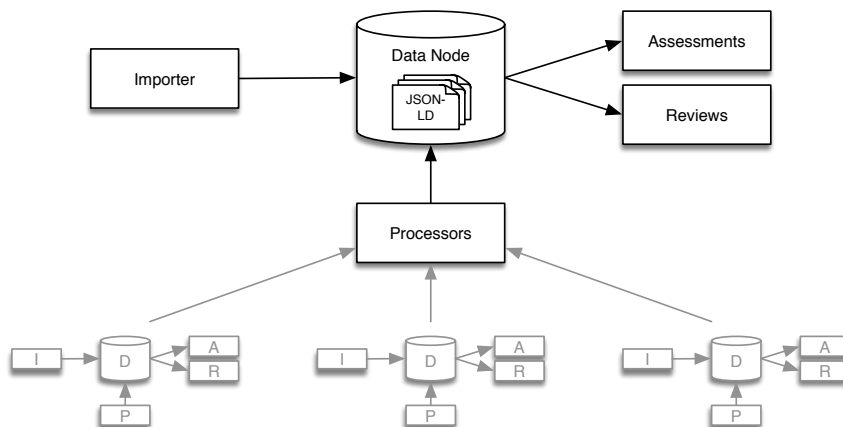


FIGURE 5.1 | Key components of the Semantification Architecture.

All elements of the quintuple except the data nodes are optional. A semantification system can be built by combining multiple modules to form a complete process where each module encapsulates specialized functionality for a certain process step.

The interconnection of the encapsulated functionalities is realized through the data nodes. All data nodes are stored in a common schema-less data base (NoSQL) and are from there accessible from all modules. This way, the output of one module can be used as data source for another module which itself can produce new data nodes and so on. Additionally the usage of a schema-less NoSQL data base ensures the extensibility of a system, as new data can be stored without constraints.

The interconnection of semantification modules into a holistic process is explained by the example of a 5-STAR semantification for PDF documents. Starting with a module that imports PDF documents, a couple of semantification steps are applied that finally yield 5-STAR Linkable Technical Documents. The process is realized using the following semantification modules.

1. **PDF Import:** An importer processes raw PDF documents and stores them as data nodes.

2. **PDF Document Layout Analysis:** A processor applies Document Layout Analysis methods to the imported data nodes. The processor stores its results in the respective data nodes.
3. **PDF Logical Document Structure Recovery:** An additional processor exploits externally provided semantification knowledge to perform Logical Document Structure Recovery task. This processor creates new data nodes that represent the recovered document micro structures like headlines or paragraphs.
4. **Macro Structure Recovery:** A subsequent processor exploits the data nodes created by the preceding processor. The data nodes representing head lines are used to recover the macro structure hierarchy of the document. On basis of the recovered macro structure the processor represents new data nodes, s.t. chapter, section and subsection is represented by a respective node. The processor is based on the methods described in Section 4.4.3.
5. **Term Matching:** A term matching processor configured with a list of relevant terms scans the data nodes created by the preceding processor for term occurrences. Discovered occurrences are stored as complementary data nodes. These complementary data nodes have a bi-directional connection to the data nodes representing chapters, sections, or subsections.
6. **Subject Indexing:** A subject indexing processor accesses the data nodes representing chapters, sections, or subsections and the linked term match data nodes. Based on the information it determines topics for the segments and stores them as new/complementary data. The processor alternatively uses the Subject Indexing methods described in Section 4.7.3 and Section 4.7.4.
7. **5-STAR Assessment:** An assessment component visualizes the subject indexing result, e.g. highlights nodes with many or few subject annotations.
8. **Subject Indexing:** Based on the assessment, the parametrization of step 5 may be revised and step 5 repeated. With stored provenance information multiple outcomes can be compared and the most appropriate one selected.
9. **5-STAR Review:** A review component allows to manually edit subject annotations, e.g. remove unnecessary or add missing subjects respectively.

The (intermediate) results, namely macro structures (modules), term matches and annotated subjects can be exported for subsequent usage in other systems. Please note that only the first three processing steps are specific for documents in PDF format and aim on producing a format independent data node representation of a PDF document. The remaining modules are generally applicable to data nodes important from any source formats.

### 5.3.3 Implementation

CAPLAN is a web application written in Java using the popular framework Google Web Toolkit<sup>5</sup>. The software runs as web application in a servlet container like Apache Tomcat<sup>6</sup> and a Java JRE<sup>7</sup> in version 8. The server can be installed on all operating systems for which the required JRE is available. Users need a web-browser in order

---

<sup>5</sup><http://www.gwtproject.org/>

<sup>6</sup><http://tomcat.apache.org/>

<sup>7</sup><http://www.oracle.com/technetwork/java/index.html>

to access the web application. The application is currently optimized for usage with browsers based on WebKit<sup>8</sup> like Google Chrome<sup>9</sup>. The following sections first give some implementation details regarding the data model and the module mechanism. Then an overview of the Graphical User Interface illustrates the usage of CAPLAN.

### Data Model

The complete architecture builds upon a very flexible schema-less data model. The data model is implemented on basis of the document-oriented NoSQL data base MongoDB<sup>10</sup>, where documents are the basic storage entity. In order to abstract from the concrete NoSQL implementation Eclipse TopLink<sup>11</sup> is used for object-relational mapping on the basis of the Java Persistence API (JPA)<sup>12</sup>. The data nodes are required to be compliant with JSON-LD [186] as storage format which is standardized, light-weight, well-supported in common data base systems, and allows to use explicit semantics. The availability of JSON-LD also allows to export (intermediate) results as standardized ontologies [89, 31]. Furthermore, JSON(-LD) is compatible with common high performance data bases that work upon Apache Hadoop, e.g. MapR-DB. Importers  $I$  and processors  $P$  have to enhance the JSON-LD documents by provenance information based on the PROV-O [116] ontology.

### Module Mechanism

The architecture is based upon the idea that a semantification system can be composed of modules that encapsulate specialized functionality. Besides a description of the data nodes (if appropriate as JSON-LD context), a module can define importers  $I$ , processors  $P$ , assessments  $A$ , and reviews  $R$ . For the integration in the framework each of these components must provide specific information. Additionally, each component might define additional parameters that are necessary for configuration. Therefore, the Java Plug-In Framework JPF<sup>13</sup>, an OSGi-like [151] plugin framework, is used.

Considering the scalability requirements modules should also be able to report whether they are capable of running in high performance environments. Therefore, modules should express their high performance capability in their plugin definitions. If they claim to be high performance capable, they are required to realize their functionality using methods of a high performance computing framework like Apache Spark [134] or Apache Flink [1].

### Graphical User Interface (GUI)

As one requirement is a high accessibility for non-experts the framework provides a graphical user interface. The graphical user interface guides users through existing semantification processes and allow for the creation of new/customized processes. Therefore, some components of the modules like importers or processors are presented in a standardized way to allow the configuration by the user. Other components like assessments or reviews require a specialized user interface. Hence, these components must also provide user interface definitions as part of a module.

---

<sup>8</sup><https://webkit.org/>

<sup>9</sup><https://www.google.de/chrome/>

<sup>10</sup><https://www.mongodb.com/>

<sup>11</sup><http://www.eclipse.org/eclipselink/>

<sup>12</sup><https://docs.oracle.com/javase/6/tutorial/doc/bnbpz.html>

<sup>13</sup><http://jpf.sourceforge.net/>

## Data Node Types

A semantification process usually has to cope with different resources ranging from terms that are concealed in term data bases over concepts from a Linked Enterprise Data graph to actual documents. In order to structure the semantification process the graphical user interface of CAPLAN<sup>14</sup> separates the semantification tasks according to the underlying node type. Figure 5.2 gives an overview of the different node types

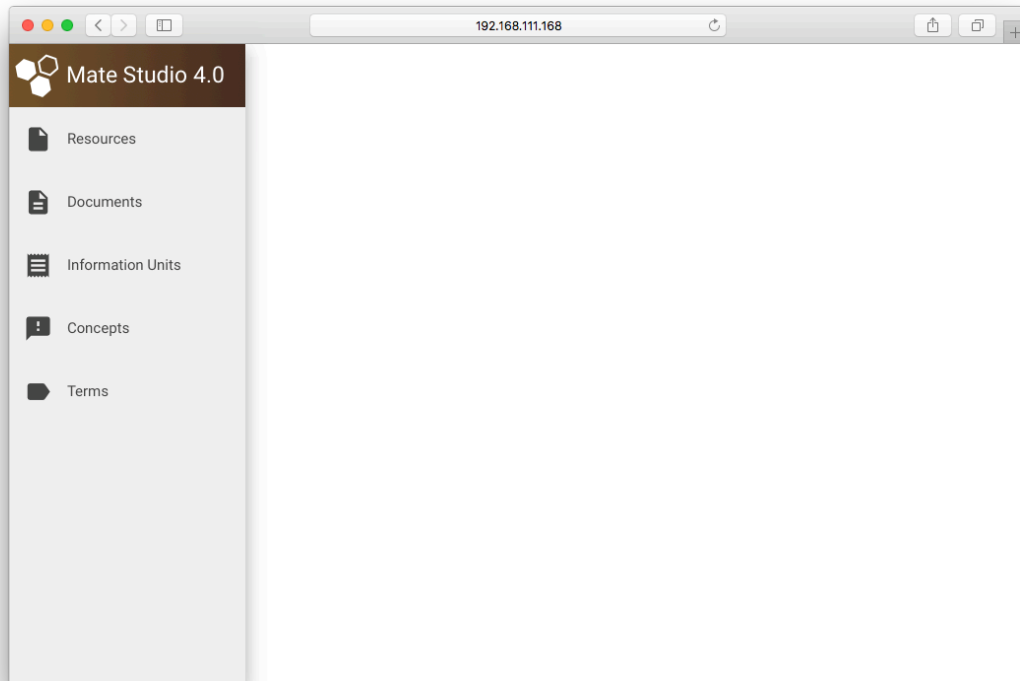


FIGURE 5.2 | CAPLAN: Overview of supported data node types.

in CAPLAN. At the time of writing supported node types in CAPLAN are:

- **Resources:** All kinds of physical resources, normally files in a file system. Resource nodes are used to handle basic tasks like downloading, moving, or cleaning files. An example for a resource is a physical PDF file stored in a local or remote file system.
- **Documents:** Resources that actually represent documents get imported as document nodes. Document Nodes already abstract from the actual file format and instead use a generic representation. This representation gives access to the underlying text and (hierarchical) relations to other document nodes. This way hierarchies of macro structures (chapters) can be represented.
- **Information Units:** Information Units are closely related to document nodes and usually represent a reasonable combination of multiple document nodes. A typical application example is a semantification process that needs to combine all language variants of a chapter to a single information unit.

<sup>14</sup>The ideas of CAPLAN have been implemented in a commercial product of denkbare GmbH that is called *Mate Studio 4.0*. The name is referring to an accompanying semantic information system called *Service Mate*.



- **Concepts:** Concepts represent nodes from a Linked Enterprise Data graph or more generally an available ontology. CAPLAN allows to import concepts from different sources and makes them available to other processing steps (e.g. Subject Indexing modules).
- **Terms:** Analogous to concept nodes CAPLAN also supports the representation of terms. In contrast to concepts they do not necessarily need to be identifiable with an URI. Terms can be imported from different sources and are then available for other processing steps. A typical usage example is an Entity Recognition processor that identifies all occurrences of a term in document nodes.

## Pipelines and Summaries

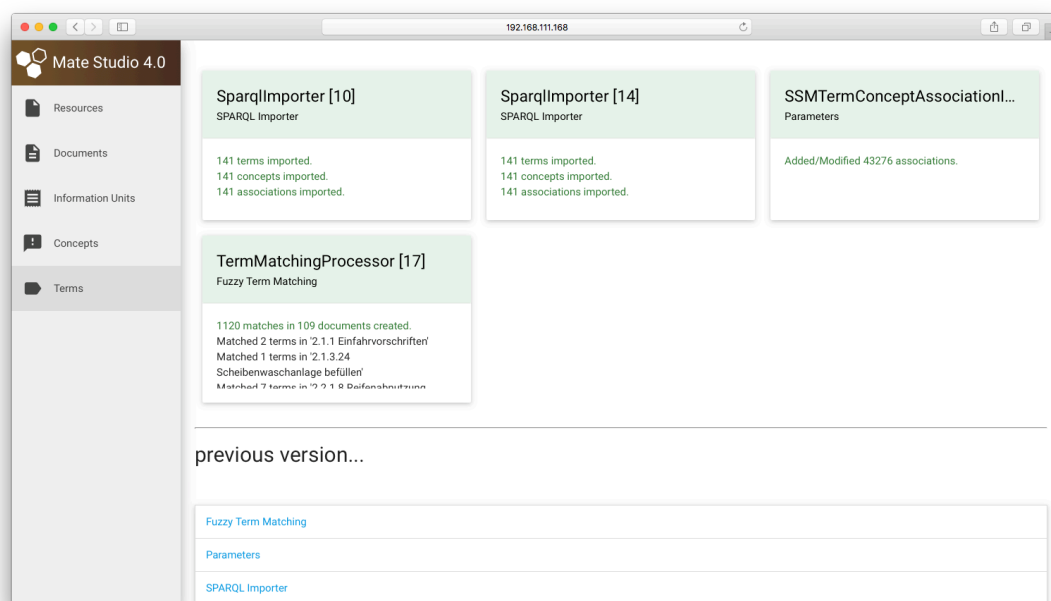


FIGURE 5.3 | CAPLAN: Modules handling Term Data Nodes.

The views of the node types follow the same basic structure. In the upper part of the view the pipeline is presented in forms of configured modules. The lower part of the view provides functionalities to extend the current pipeline with additional modules. Each semantification module is represented by a respective card. If the pipeline has already been executed the status of the last processing is indicated. Figure 5.3 shows a pipeline for the import of terms and a subsequent Entity Recognition (term matching) step based on these terms.

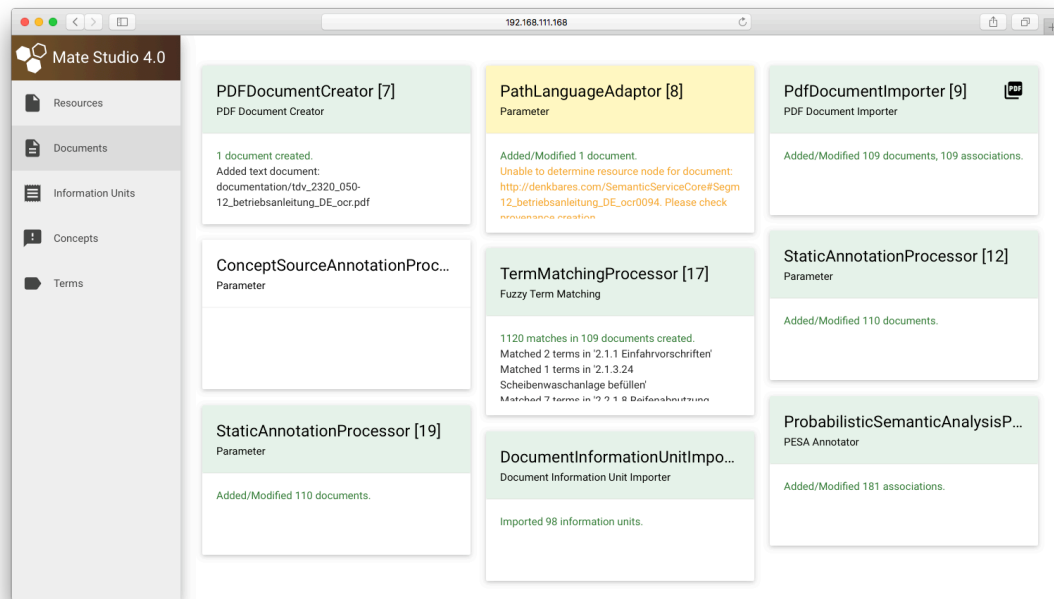


FIGURE 5.4 | CAPLAN: Modules handling Document Data Nodes.

Green cards indicate that the module has been executed successfully. Cards that have a yellow background color indicate that the module has been executed but minor inconsistencies have been discovered. Figure 5.4 shows a pipeline that is processing document nodes that originated from PDF documents. The pipeline contains a module that tries to determine the language of a document for which an inconsistency has been detected during the last run.

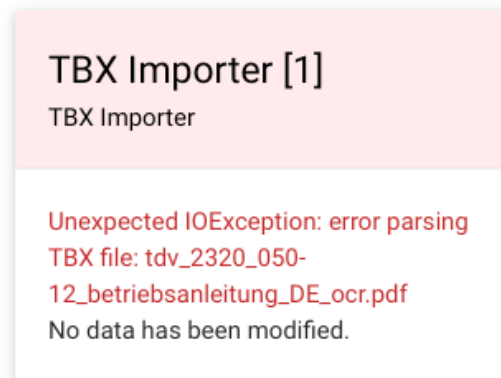


FIGURE 5.5 | CAPLAN: Error in Semantification Module.

Red cards indicate that the processing of the respective semantification module failed. Detailed messages allow the user to examine and evaluate the discovered errors. Figure 5.5 shows an extended term processing pipeline where an error occurred while trying to import terms from a TBX file.

## Module Library

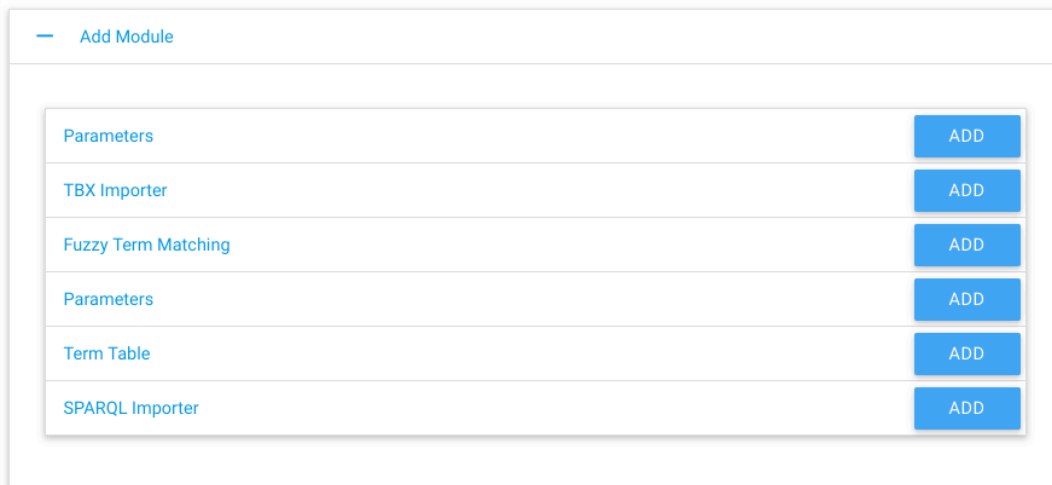


FIGURE 5.6 | CAPLAN: Library of Semantification Modules.

The assembly of semantification pipelines is an important aspect for designing semantification projects. Therefore, CAPLAN exploits the plugin information of each semantification module on the Java classpath to provide the user an always up-to-date catalogue of available semantification modules. The catalogue is filtered according to the current node type. Figure 5.6 shows the catalogue of available modules for handling term nodes.

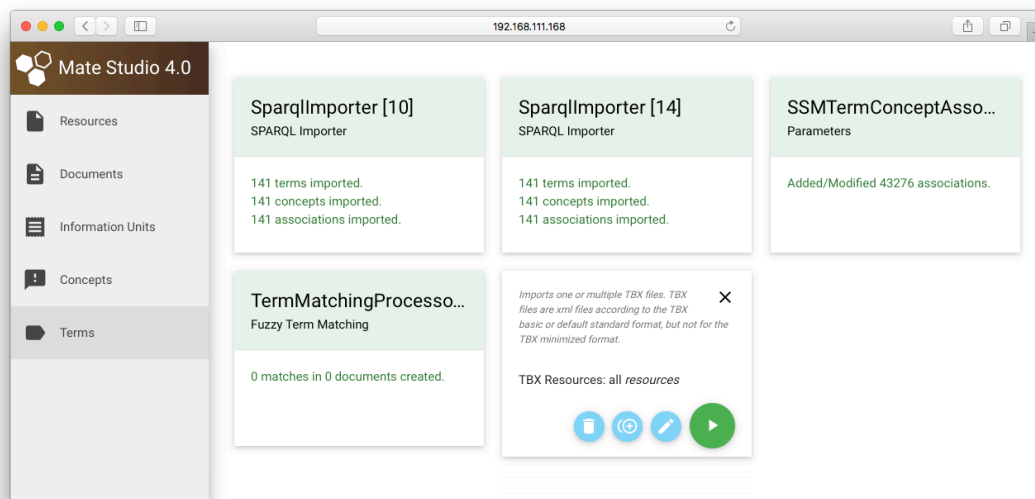


FIGURE 5.7 | CAPLAN: Configuration of a Semantification Module.

The catalogue allows to quickly add a semantification module to the current pipeline. Therefore, the user has to select the respective add button of a listed module. Afterwards the module gets automatically added to the current pipeline. Like all other modules the newly added module can be configured by the user (see Figure 5.7). Basic configuration functionalities comprise the removal of the respective module from the pipeline, the duplication of the respective module from the pipeline, as well as the parametrization of the module.

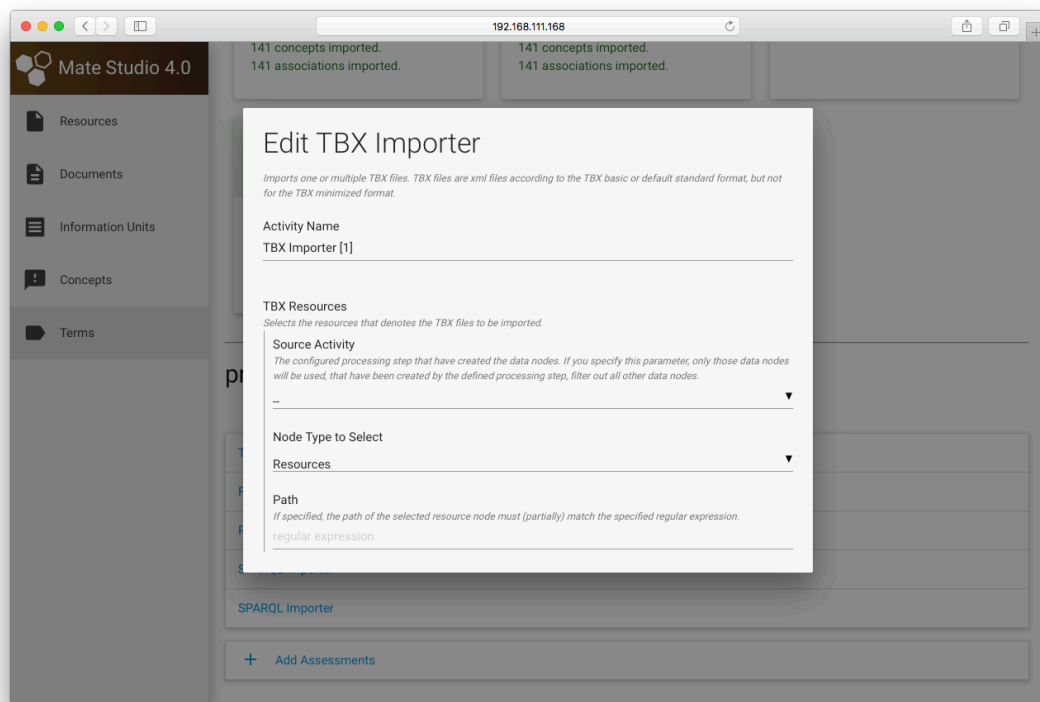


FIGURE 5.8 | CAPLAN: Parametrization of a Semantification Module.

For the parametrization of a module a generic view is available that reads all configuration parameters as well as allowed parameter values from the module definition. Figure 5.8 shows the configuration dialog for a module that imports terms from TBX files.

## 5.4 Developing Semantification Knowledge: TEKNO Studio

The cloud-based semantification architecture CAPLAN enables users to realize semantification projects. However, for some processing steps domain-specific knowledge is required. While the knowledge acquisition is trivial for already formalized knowledge sources like ontologies it is harder for rather unstructured base data. This is especially true for classification knowledge that is required for the 2-STAR semantification. The 2-STAR semantification aims on detecting and classifying micro structures. Section 4.4.3 proposed a classification approach based on Set-Covering Models as this (1) exploits the fact that technical documentation is rather constant with respect to formatting and (2) handles uncertainty which might occur due to inconsistent usage of formatting rules.

Although Set-Covering Models are a catchy knowledge representation compared to other forms, it remains hard to define concrete Set-Covering relations. The definition of Set-Covering relations usually requires to define specific values for features. In the context of formatting information this might be exact font size or font name or height and width of text blocks. The PDF format does not give direct access to these information. Hence, TEKNO Studio, a tool that supports the creation of Set-Covering Models from PDF documents, has been developed.

TEKNO Studio is based on several existing libraries. The library d3web<sup>15</sup> provides implementations of strong-problem solving methods, the basic object model for Set-Covering models, and the corresponding classification mechanism. The library LA-PDFText<sup>16</sup> is able to identify contiguous text blocks in PDF documents and is employed in TEKNO Studio for blockification purposes. LA-PDFText and the visualization component of TEKNO Studio make usage of functionalities of the JPedal library<sup>17</sup>, which supports the handling and visualization of PDF documents in Java. TEKNO Studio provides a graphical user interface that is based on JavaFX<sup>18</sup> and follows the Model-View-Presenter architectural pattern.

#### 5.4.1 Workflow

TEKNO Studio is a tool that accompanies the CAPLAN semantification environment for developing required classification knowledge. The combined workflow of CAPLAN and TEKNO Studio can be sketched as follows:

1. **Knowledge Acquisition:** The knowledge acquisition, i.e. the definition of Set-Covering Models for a corpus of technical documents is performed in TEKNO Studio.
2. **Knowledge Export:** The acquired knowledge gets exported in form of a compiled knowledge base in d3web format.
3. **Knowledge Import:** CAPLAN contains a resource module that is able to import d3web knowledge bases. The imported knowledge bases are then represented as resource nodes in CAPLAN.
4. **Blockification:** CAPLAN contains a blockification module for documents in PDF format that use the same blockification mechanism as TEKNO studio. The blockification mechanism is based on LA-PDFText as stated before. The discovered blocks are represented as micro structures (document nodes) in CAPLAN.
5. **Classification:** CAPLAN contains a classification module for micro structures. This classification module loads the classification knowledge from the already imported d3web knowledge base. Then each block gets classified in a d3web problem solving session according to the knowledge base. Therefore, CAPLAN's classification module creates findings (answers to questions in d3web) in each problem solving session that represent the formatting features of the micro structure in focus. Then d3web uses the Set-Covering knowledge to determine the most appropriate class(es) for the block.

---

<sup>15</sup><https://www.d3web.de>

<sup>16</sup><https://github.com/BMKEG/lapdftext>

<sup>17</sup><https://www.idrsolutions.com/jpedal/>

<sup>18</sup><http://www.oracle.com/technetwork/java/javafx/overview/index.html>

### 5.4.2 Implementation

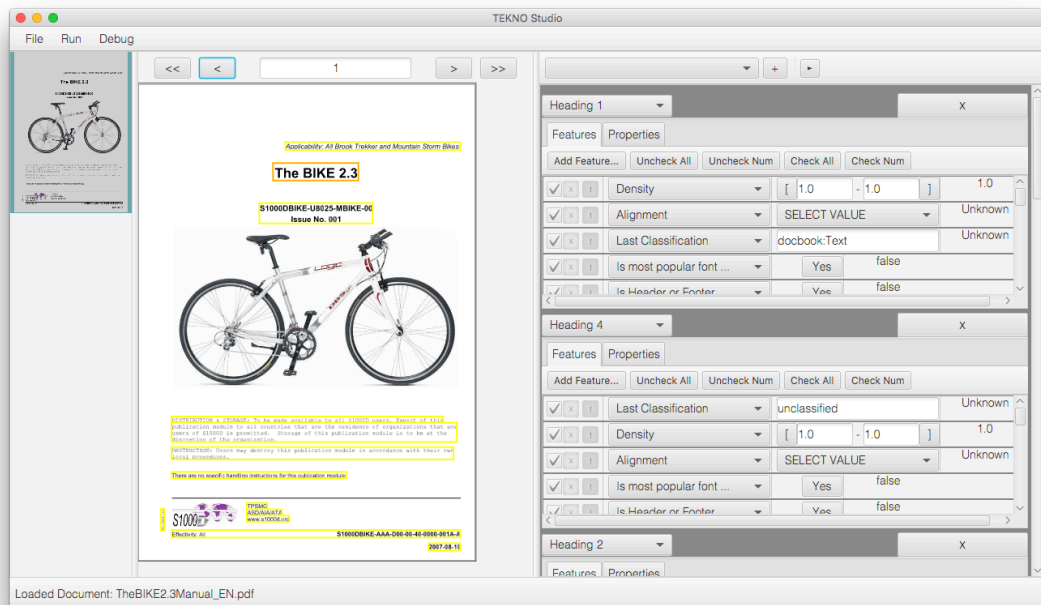


FIGURE 5.9 | TEKNO: Blockification and Block Selection.

TEKNO Studio operates with the classification object in focus, i.e. the whole functionality is focused on the technical document. Hence, the main view is filled with a preview of the currently loaded document. Buttons allow the page-wise navigation in the document. On the left side a thumbnail view provides quick access to other documents. TEKNO Studio automatically performs the blockification task for a loaded document. Discovered blocks on the current page get indicated using yellow boxes (see Figure 5.9). All blocks can be selected by the user by performing a left click. A right click on a block deselects it. Selecting multiple blocks is possible and usually reasonable. Selected blocks are indicated by orange boxes (see Figure 5.9).

TEKNO Studio provides the functionality to automatically create a Set-Covering Model from the currently selected blocks. Therefore, the user has to choose one of the project-specific classes from the drop-down menu on the right side of the graphical user interface (see Figure 5.10). When a class has been selected TEKNO automatically determines a model with reasonable Set-Covering relations for the selected blocks. The automatically created Set-Covering Model can be manually edited by the user. Therefore, existing Set-Covering relations can be adjusted or removed. The addition of new Set-Covering relations is also possible. Features are predefined and need to be selected from a drop-down menu. Feature values can be defined by the user. It is to note that TEKNO Studio automatically performs a consistency check for defined feature values. For debugging purposes the feature values of the currently selected blocks get displayed next to each set-covering relation. Each change automatically results in a recompilation of the underlying knowledge base. Thus, manual saving operations by the user are not necessary.

TEKNO Studio provides the user the possibility to test the defined classification knowledge against the currently loaded document. The classification gets started using a dedicated run button next to the drop-down menu on the right side of the

graphical user interface. TEKNO Studio then performs the same classification procedure as the respective CAPLAN processor, i.e. starting a d3web problem solving session for each block and using the results as class assignment. All blocks that were classified are indicated by a red surrounding box that is labeled with the respective class.

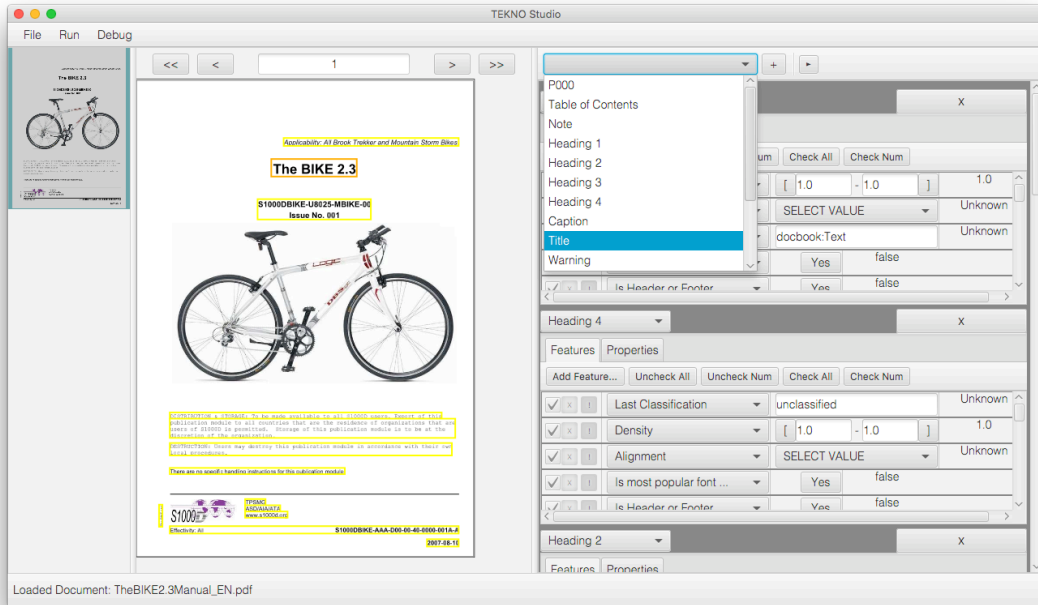


FIGURE 5.10 | TEKNO: Create a Set-Covering Model.

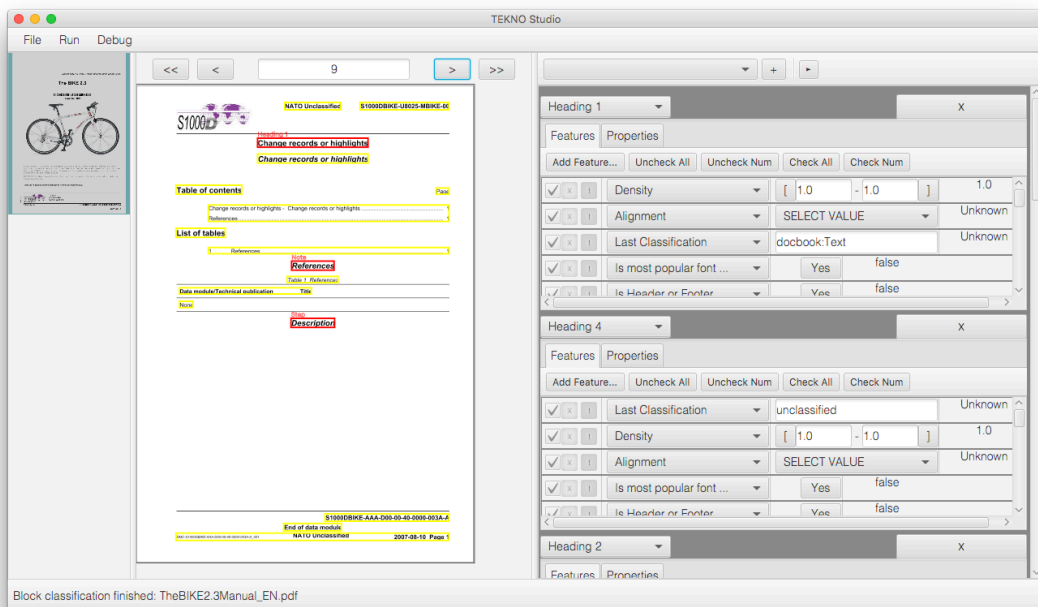


FIGURE 5.11 | TEKNO: Preview of classification results.

## 5.5 Reviewing Semantification Results: Review Tool

The processing of technical documents using CAPLAN and TEKNO finally yields 5-STAR Linkable Technical Documents. However, the processing results might contain inconsistent, missing, or wrong links to the Linked Enterprise Data graph. Depending on the required quality criteria a manual review of the processing results is necessary. Therefore, a dedicated review tool for 5-STAR semantification results has been developed (See Figure 5.12).

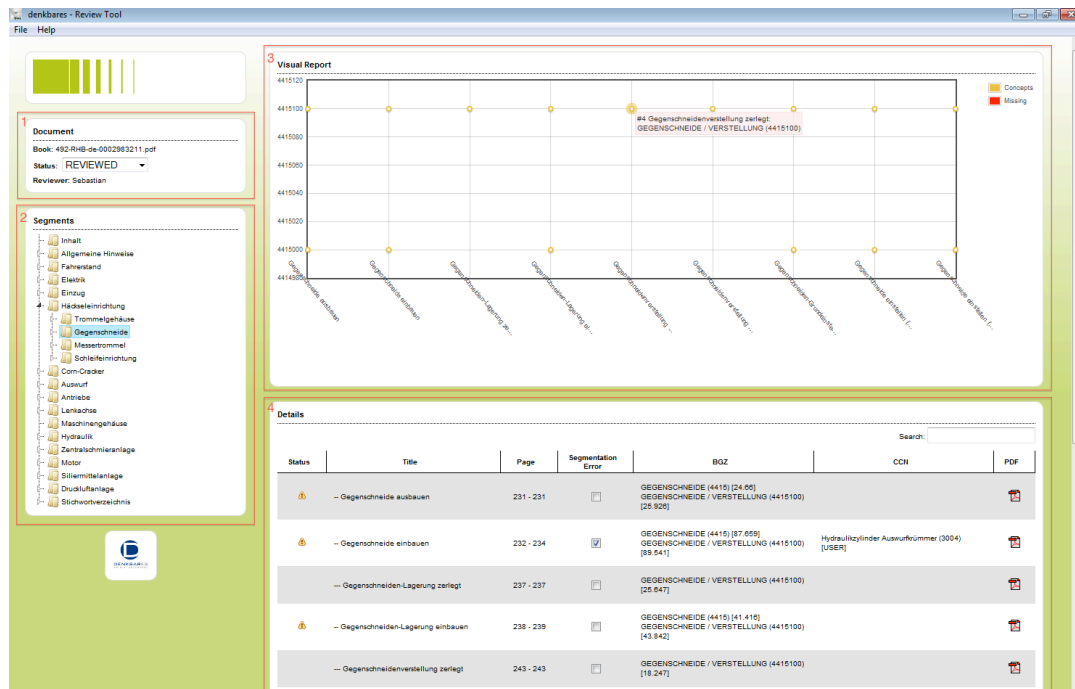


FIGURE 5.12 | Review Tool: Overview.

The review tool is able to load 5-STAR semantification results in RDF format, which is the standard output of CAPLAN. The review tool is designed as web application embedded in a desktop application using the servlet container Jetty<sup>19</sup>. On the server side Java servlets provide the business logic. The client side code is based on standard HTML, CSS, and JavaScript.

The review tool is assembled from four views:

1. **Status:** The Status View provides basic information about the document, the review status, and the reviewer.
2. **Macro Structure Hierarchy:** This view displays the recovered macro structure hierarchy of the currently reviewed document.
3. **Visual Report:** A visual report allows to quickly identify wrong, inconsistent or missing links.
4. **Details:** This view allows the reviewer to make changes to the respective semantification results.

In the following the single views are described in more detail.

<sup>19</sup><https://www.eclipse.org/jetty/>



A central element of the 5-STAR review tool is a tree view (see Figure 5.13) that visualizes the recovered macro structure hierarchy. The user can navigate through this hierarchy by clicking the respective tree view elements. A selection of an element of the tree view updates the contents of the visual report and the details view. The review tool automatically saves the last selection in the review tool. This enables the user to continue his work after the review tool had been closed.

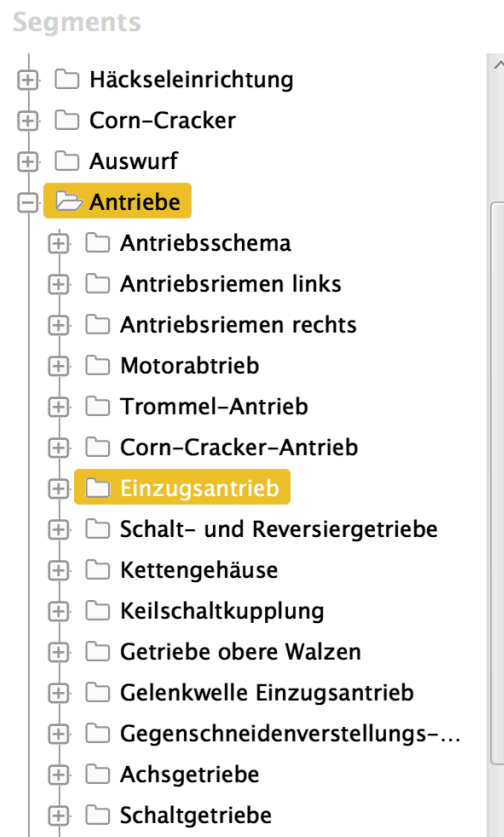


FIGURE 5.13 | Review Tool: Macro Structure Hierarchy.

Reviewing every single data entry in the recovered 5-STAR data would require a massive amount of time. Therefore, the review tool provides a visual report. This visual report displays the semantic similarity of the annotated concepts from the underlying Linked Enterprise Data graph. As concept annotations usually follow characteristic patterns a reviewer can quickly identify outliers.

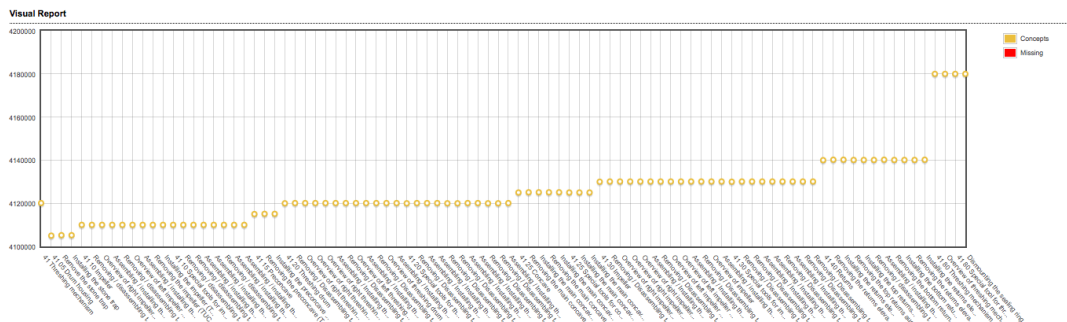


FIGURE 5.14 | Review Tool: Visual Report.

Figure 5.14 shows a visual report that illustrates a characteristic staircase pattern. The pattern is based on the fact that in technical documents successive chapters have the same or similar topics, e.g. chapters that describe the disassembly, the repair and the assembly of a component. Links (concept annotations) that violate the pattern are candidates that might need manual review and adaptation. For the manual adaptation of single entries a details view provides the relevant information. A click on a data point in the visual report selects the respective entry in the details view.

Status	Title	Page	Segmentation Error	BGZ	CCN	PDF
	-- Gegenschneide ausbauen	231 - 231	<input type="checkbox"/>	GEGENSCHNEIDE (4415) [24.68] GEGENSCHNEIDE / VERSTELLUNG (4415100) [25.928]		
	Gegenschneide einbauen	232 - 234	<input checked="" type="checkbox"/>		Hydraulikzylinder Auswurfkrümmer (3004) [USER]	

FIGURE 5.15 | Review Tool: Editing Macro Structure Titles.

The details view allows to edit all relevant 5-STAR information. Figure 5.15 shows the editing of a macro structure title. Figure 5.16 shows the actual editing of semantic annotations (links). Existing annotations can be removed. For the addition of new annotations a Semantic Autocompletion [95] component supports the user by selecting the correct concept from the Linked Enterprise Data graph. The details view additionally provides quick access to the original document, i.e. a click on the document icon opens the document at the position that is described by the macro structure.

Status	Title	Page	Segmentation Error	BGZ	CCN	PDF
	-- Gegenschneide ausbauen	231 - 231	<input type="checkbox"/>	GEGENSCHNEIDE (4415) [24.68] GEGENSCHNEIDE / VERSTELLUNG (4415100) [25.928]		
	-- Gegenschneide einbauen	232 - 234	<input checked="" type="checkbox"/>	GEGENSCHNEIDE (4415) [87.659] ✕ GEGENSCHNEIDE / VERSTELLUNG ✕ (4415100) [89.541] gegensch KABELSATZ GEGENSCHNEIDENVERSTELLUNG (4415860) GEGENSCHNEIDE / VERSTELLUNG (4415100) MODUL GEGENSCHNEIDE (4415050) GEGENSCHNEIDE (4415)	Hydraulikzylinder Auswurfkrümmer (3004) [USER]	
	--- Gegenschneiden-Lagerung zerlegt	237 - 237	<input type="checkbox"/>			
	-- Gegenschneiden-Lagerung einbauen	238 - 239	<input type="checkbox"/>			

FIGURE 5.16 | Review Tool: Annotation Editing supported by Semantic Typing.

## 5.6 Summary and Contributions

This chapter presented a tool box that supports the 5-STAR semantification approach. The tool box consists of three applications. The most important tool is CAPLAN which is a cloud-based semantification architecture that aims on enabling the accessible and scalable realization of semantification projects. Therefore, it provides a clean graphical user interface that is intended to be used by non-expert users. Users can assemble semantification pipelines from a library of existing modules. A plugin mechanism allows for the easy extension of CAPLAN with new semantification modules.

The second application is TEKNO Studio. Unlike CAPLAN that tackles the whole semantification process TEKNO Studio is a tool that is especially designed to support the knowledge acquisition for the 2-STAR semantification step. Therefore,

---

an interactive environment displays PDF documents and allows to create and debug classification knowledge in forms of Set-Covering Models.

Finally, a dedicated review tool allows to review and edit the results of the 5-STAR semantification process, i.e. results created by the CAPLAN application. The review tool is designed to support a targeted review by guiding a human reviewer to potentially inconsistent data entries. Therefore, the semantic similarity of semantic annotations is visualized in a report that allows to easily identify outlier concepts. Elaborated editing features allows the user to quickly edit inconsistent entries.



## Chapter 6

# Hands On: Semantification by Example

The previous chapters introduced the 5-STAR semantification process (cf. Chapter 4) and its implementation in TEKNO Studio and CAPLAN (cf. Chapter 5). This chapter covers an extended example that describes details of the particular semantification steps and thus serves as a hands on guide for semantification projects. The detailed description of each semantification step contains source and input data as well as used tools and required configurations. Additionally, the generated output will be listed for each processing step.

## 6.1 Source Data and Semantification Goal

This example operates upon a minimum but representative partition of the S1000D data set (cf. Section 7.2). Hence, a single chapter (“Steering – Description of how it is made”) of the S1000D Bike manual is semantified in the course of this chapter. The respective chapter (see Figure 6.1) is available in PDF format and contains a variety of micro structures (headlines on different levels, table of contents, etc.) and macro structures (chapter, sections, and subsections). However, the PDF document does not contain any semantic information, i.e., the respective structures are not accessible and not semantically annotated. The goal of this example is to show the successive transformation of these locked up sections (“Steering”) and subsections (“Handlebar”, “Headset”, and “Stem”) into information typed and semantically annotated modules (information units). The resulting improved accessibility is demonstrated using the state-of-the-art information system “Service Mate”.



FIGURE 6.1 | Hands On: Source PDF.

## 6.2 1-STAR: Document Layout Analysis

The TEKNO ontology describes document structures on different levels of detail (nano, micro, and macro structures) that successively improve the accessibility of documents (cf. Section 3.2). These structures are recovered bottom up, beginning with nano structures. As the PDF format does not provide structured access to any kind of document components the first step comprises the recovery of nano structures (tokens) and untyped micro structures (blocks). The recovery is realized using methods from Document Layout Analysis (cf. Section 4.3.3). As stated in Section 4.3.6 a variety of utilities for Document Layout Analysis exists. This example uses pdf2xml [52] that is available for all major operating systems as command line utility. The pdf2xml utility converts the sample PDF document to an XML representation. Listing 6.1 shows the respective command and the recommended configuration, which exports the full name of fonts and already clusters single text tokens in blocks.

```
pdf2xml -fullFontName -blocks <pdf> <xml>
```

LISTING 6.1 | Hands On: Executing pdf2xml.

The command generates a file in pdf2xml XML format (cf. Section 4.3.4). This file contains information for accessing pages, blocks and tokens (see Listing 6.2). Figure 6.2 shows an alignment with elements from the original document.

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCUMENT>
...
<PAGE width="595" height="842" number="105" id="p2">
  <BLOCK id="p105_b18">
    <TEXT id="p105_t24" x="70.87" y="532.698">
      <TOKEN id="p105_w52" font-name="helvetica-bold"
        symbolic="yes" fixed-bold="yes" italic="no"
        font-size="14" font-color="#000000"
        x="70.87" y="532.698">1</TOKEN>
    </TEXT>
    <TEXT id="p105_t25" x="127.56" y="532.698">
      <TOKEN id="p105_w53" font-name="helvetica-bold"
        symbolic="yes" fixed-bold="yes" italic="no"
        font-size="14" font-color="#000000"
        x="127.56" y="532.698">Steering</TOKEN>
    </TEXT>
  </BLOCK>
  <BLOCK id="p105_b19">
    <TEXT x="127.56" y="562.4" id="p105_t26">
      <TOKEN id="p105_w54" font-name="helvetica"
        bold="no" italic="no" font-size="10"
        font-color="#000000" x="127.56"
        y="562.4">The</TOKEN>
      <TOKEN id="p105_w55" font-name="helvetica"
        bold="no" italic="no" font-size="10"
        font-color="#000000" x="147.57"
        y="562.4">steering</TOKEN>
    ...
  </TEXT>
</BLOCK>
...
</PAGE>
</DOCUMENT>
```

LISTING 6.2 | Hands On: pdf2xml XML (1-STAR).

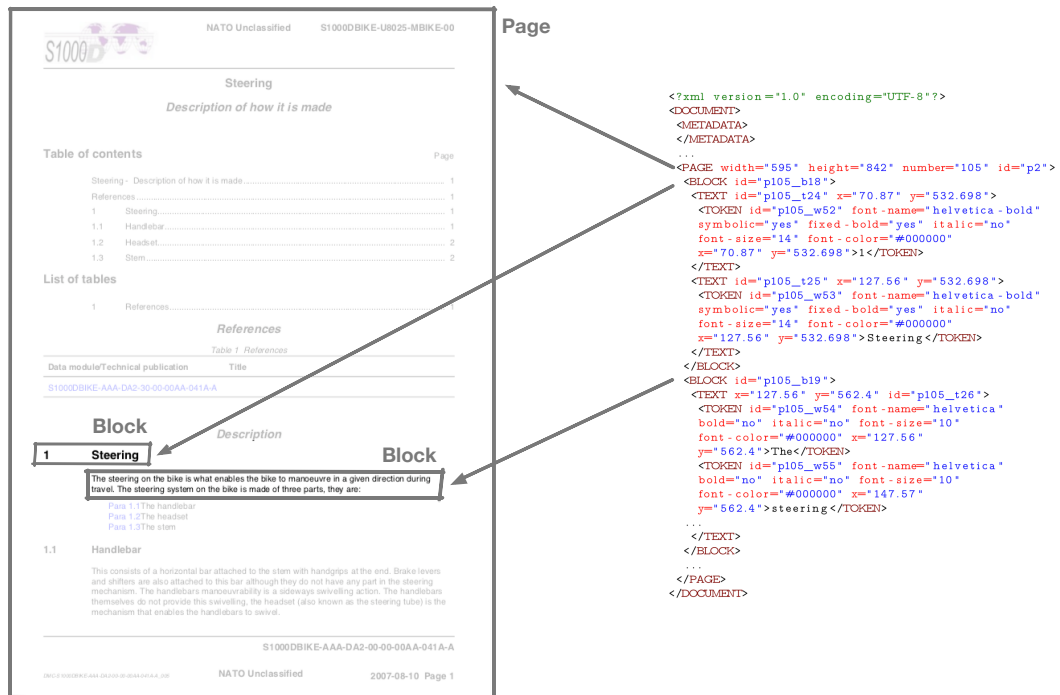


FIGURE 6.2 | Hands On: 1-STAR results.

## 6.3 2-STAR: Logical Document Structure Recovery

The 1-STAR semantification described in the previous section resulted in an XML representation of the source document that already gives access to basic document structures like pages, untyped blocks and single tokens. However, a targeted access to certain document structures, that might be used for improving textual search indexes or information extraction tasks (cf. Section 3.2), is not yet possible. This requires the typing/classification of yet untyped structures.

This work described a knowledge-based approach for Logical Document Structure Recovery (cf. Section 4.4.3) that focuses on the classification of document structures using Set Covering knowledge. The described approach relies on the homogeneity of technical documents with respect to structuring and formatting, i.e., defining classification knowledge using a small amount of example structures and then using the acquired knowledge to classify all structures in a corpus. Therefore, the interactive knowledge acquisition tool TEKNO Studio (cf. Section 4.4.4 and Section 5.4) is used to define classification knowledge for relevant document structures. The following paragraphs describe the acquisition of respective classification knowledge using TEKNO Studio and discuss the resulting Set Covering models in detail.

### 6.3.1 TEKNO Studio: Knowledge Acquisition

In practice not all document structures need to be classified. Usually it is sufficient to classify the structures that are required for further processing. As the further processing aims on the recovery of macro structures, the following structure types are considered relevant for this example:

- **Heading 1**, e.g. “Steering”
- **Heading 2**, e.g. “Description of how it is made”



- **Heading 3**, e.g. “Steering”
- **Heading 4**, e.g. “Handlebar”, “Headset”, and “Stem”

In order to define classification knowledge for the aforementioned types the PDF source document and the generated XML representation (1-STAR) are used as input for TEKNO Studio. TEKNO Studio displays the loaded document page-wise and marks untyped micro structures (1-STAR information) with yellow boxes. Then, the user selects examples of the respective structures and uses TEKNO Studio to automatically generate a suggestion for a corresponding Set Covering model (cf. Section 4.4.3).

TEKNO Studio allows the user to edit and test the resulting classification knowledge. Therefore, the user simply clicks on the “run” button, which triggers the classification of all blocks in the document. Figure 6.3 and Figure 6.4 show the pages from the example document with classified blocks (depicted in red). This way, classification errors can be identified and the classification knowledge adjusted accordingly.

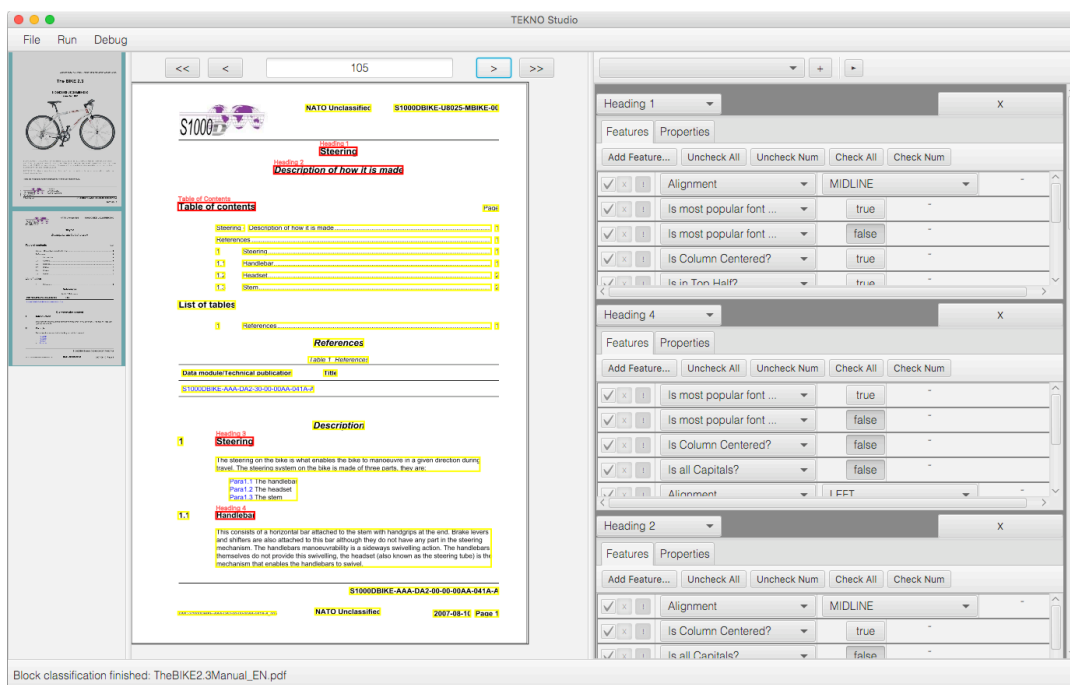


FIGURE 6.3 | Hands On: Typed Micro Structures (1/2).

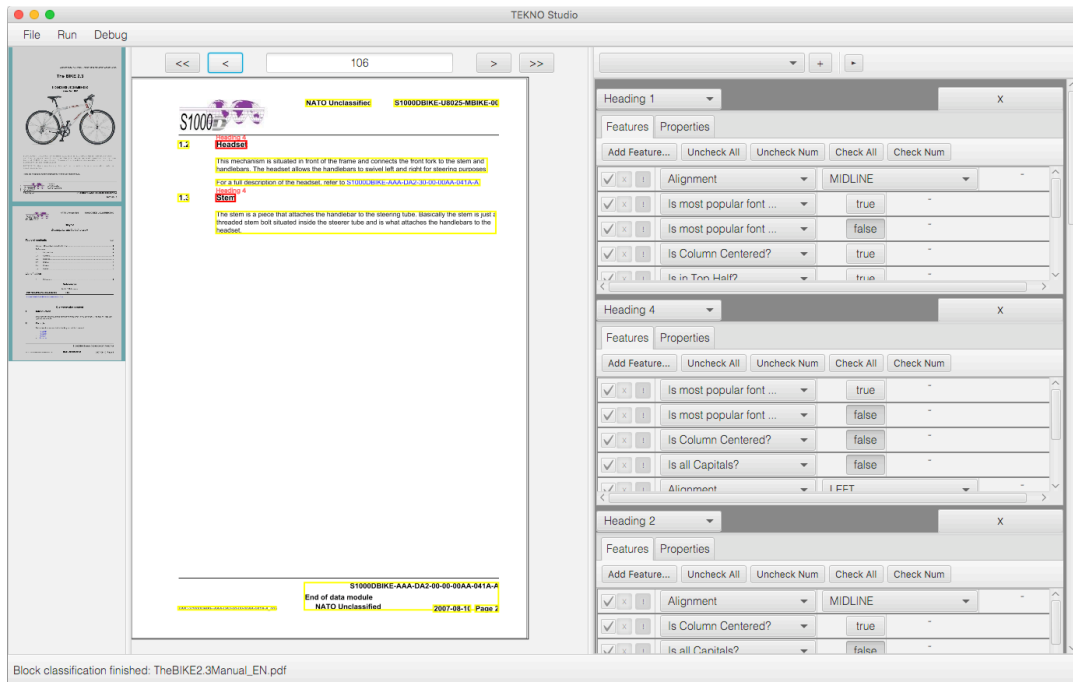


FIGURE 6.4 | Hands On: Typed Micro Structures (2/2).

### 6.3.2 Classification Knowledge

The generated Set Covering models consist of multiple Set Covering relations that use a fixed set of parameters that reflect features of document structures. The Set Covering classification in TEKNO Studio is powered by d3web<sup>1</sup>. Listing 6.3 shows a d3web question tree in KnowWE [10] markup that reflects the available features for Set Covering relations in TEKNO Studio. Most of these features operate upon formatting information, which are available from the recovered 1-STAR representation. These features are implemented as d3web questions and can be of type one choice ([ocl]), yes/no ([yn]), text ([text]), and numeric ([num]). Listings 6.4 – 6.7 show the resulting Set Covering models for the micro structure types “Heading 1”, “Heading 2”, “Heading 3”, and “Heading 4” in KnowWE markup.

<sup>1</sup><https://www.d3web.de>

```

%%Question
Alignment and Position
- Alignment [oc]
-- LEFT
-- RIGHT
-- MIDLINE
- Is Aligned with Column Boundaries? [yn]
- Is Column Centered? [yn]
- Is Outlier? [yn]
- Is in Top Half? [yn]
- Number of Line [num]
- Is Header or Footer [yn]
- X-Left [num]
- X-Right [num]
- Y-Top [num]
- Y-Bottom [num]
- Height [num]
- Width [num]
Font
- Font Name [text]
- Most Popular Font Size [num]
- Is all Capitals? [yn]
- Is most popular font modifier bold? [yn]
- Is most popular font modifier italic? [yn]
- Is most popular font in document? [yn]
- Is next most popular font in document? [yn]
Text in Chunk
- Chunk Text Length [num]
- Density [num]
- Height Difference between Chunk and Document Word [num]
- Contains first Line of Page? [yn]
- Contains last Line of Page? [yn]
- Chunk Text [text]
Document
- Page Number [num]
Classification
- Last Classification [text]
%
```

LISTING 6.3 | Hands On: Set Covering features.

```

%%CoveringList
Heading 1 {
  Alignment = MIDLINE [1.0],
  Is most popular font modifier bold? = Yes [1.0],
  Is most popular font modifier italic? = No [1.0],
  Is Column Centered? = Yes [1.0],
  Is in Top Half? = Yes [1.0],
  Density [0.91 1] [1.0],
  Most Popular Font Size [12 12] [1.0],
  Most Popular Font Size [12 12] [1.0],
  Font Name = /HELVETICA-BOLD/ [!],
  Y-Top [93 98] [!]
}
@minSupport: 0.5
%
```

LISTING 6.4 | Hands On: Set Covering model for Heading 1.

```
%%CoveringList
Heading 2 {
  Alignment = MIDLINE [1.0],
  Is Column Centered? = Yes [1.0],
  Is all Capitals? = No [1.0],
  Is in Top Half? = Yes [1.0],
  Density [0.91 1] [1.0],
  Most Popular Font Size [12 12] [1.0],
  Y-Top [122 126] [!],
  Last Classification = /Heading 1/ [!],
  Font Name = /HELVETICA-BOLDOBLIQUE/ [!]
}
@minSupport: 0.5
%
```

LISTING 6.5 | Hands On: Set Covering model for Heading 2.

```
%%CoveringList
Heading 3 {
  Is most popular font modifier bold? = Yes [1.0],
  Is most popular font modifier italic? = No [1.0],
  Is all Capitals? = No [1.0],
  Alignment = LEFT [1.0],
  Font Name = /HELVETICA-BOLD/ [!],
  Most Popular Font Size [12 12] [!],
  X-Left [127 127] [!]
}
@minSupport: 0.5
%
```

LISTING 6.6 | Hands On: Set Covering model for Heading 3.

```
%%CoveringList
Heading 4 {
  Is most popular font modifier bold? = Yes [1.0],
  Is most popular font modifier italic? = No [1.0],
  Is Column Centered? = No [1.0],
  Is all Capitals? = No [1.0],
  Alignment = LEFT [1.0],
  Font Name = /HELVETICA-BOLD/ [!],
  Most Popular Font Size [11 11] [!],
  X-Left [127 127] [!]
}
@minSupport: 0.5
%
```

LISTING 6.7 | Hands On: Set Covering model for Heading 4.

### 6.3.3 Output

The result of the classification based on Set Covering models are typed micro structures. For further processing a typed micro structure representation of the complete document needs to be exported. A variety of compatible XML formats exists (cf. Section 4.4.6). As TEKNO Studio builds upon the LAPDF-text utility [162] the results are exported in LAPDF-text XML format. Basically, this XML format gives access to pages (**Pages**), nano structures (**Word**) and micro structures (**Chunk**). Additionally, the export to this format benefits from the heuristics for reading order determination that are included in the LAPDF-text utility. Listing 4.3.4 shows an excerpt of the resulting XML file. In contrast to the 1-STAR export, the micro structures (**Chunk**)

in this XML export have an attribute `type` that reflects the type of the block. Figure 6.5 shows an alignment of the 2-STAR results with elements from the original document.

```
<Page x1="70" y1="32" x2="552" y2="802" chunkCount="45"
  pageNumber="105" wordCount="177">
  ...
  <Chunk x1="283" y1="96" x2="339" y2="108" type="Heading 1">
    <Word x1="283" y1="96" x2="339" y2="108"
      font="Helvetica - Bold" style="Bold">Steering</Word>
  </Chunk>
  <Chunk x1="214" y1="124" x2="408" y2="136" type="Heading 2">
    <Word x1="214" y1="124" x2="291" y2="136"
      font="Helvetica - BoldOblique" style="Bold">Description</Word>
    <Word x1="295" y1="124" x2="308" y2="136"
      font="helvetica - boldoblique" style="Bold">of</Word>
    <Word x1="312" y1="124" x2="340" y2="136"
      font="helvetica - boldoblique" style="Bold">how</Word>
    <Word x1="344" y1="124" x2="352" y2="136"
      font="helvetica - boldoblique" style="Bold">it</Word>
    <Word x1="356" y1="124" x2="367" y2="136"
      font="helvetica - boldoblique" style="Bold">is</Word>
    <Word x1="372" y1="124" x2="408" y2="136"
      font="helvetica - boldoblique" style="Bold">made</Word>
  </Chunk>
  ...
</Page>
```

LISTING 6.8 | Hands On: LAPDF-text XML (2-STAR).

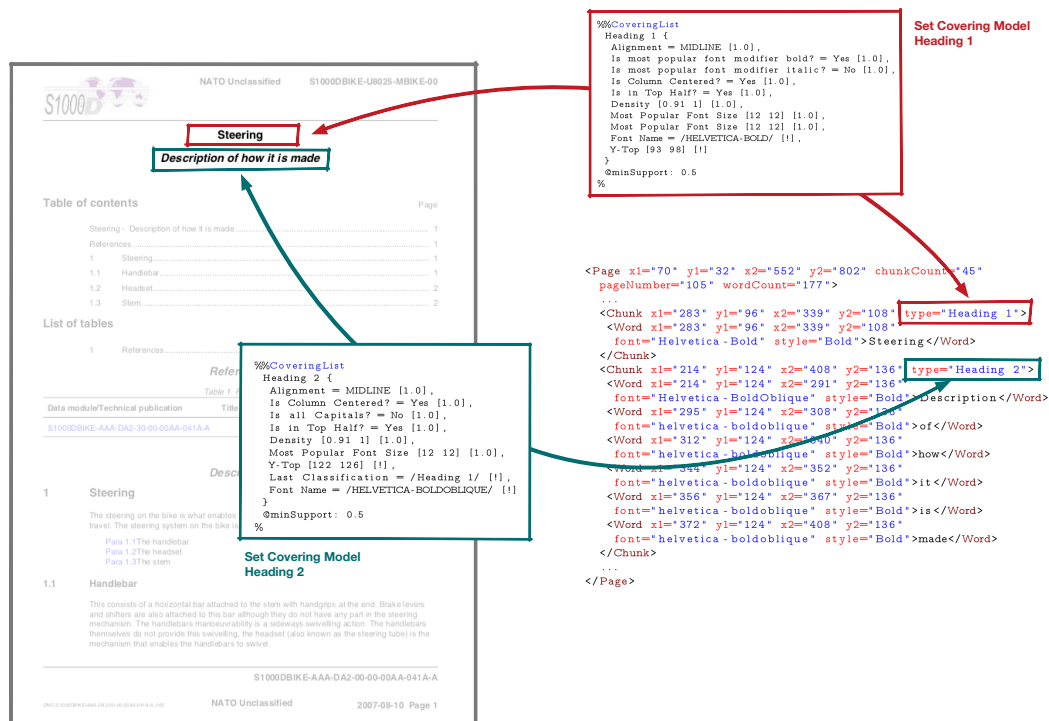


FIGURE 6.5 | Hands On: 2-STAR results.

## 6.4 3-STAR: Macro Structure Recovery

The 2-STAR semantification described in the previous section resulted in an XML representation of the source document that gives access to document structures on different levels of detail. In contrast to 1-STAR data, the structures are typed. This enables the direct access to structures in further processing steps.

The 3-STAR semantification aims on recovering the complete macro structure of a document, i.e., a hierarchy of chapters, sections, and subsections. As information about the macro structure of a document is usually not available from source documents, this work described a recovery approach (cf. Section 4.5.3) that exploits classified micro structure information (2-STAR) to algorithmically determine a macro structure hierarchy. Headlines on different levels usually mark the start and end of macro structures and are thus primarily considered by the recovery approach.

The previous section described the classification of micro structures using Set Covering models, which resulted in the availability of typed micro structures that represent headlines on different levels. These typed micro structures are taken as input for a CAPLAN processor (cf. Section 5.3) that implements the aforementioned macro structure recovery approach. The processor creates macro structure instances according to the underlying data model. Figure 6.6 shows the resulting 3-STAR macro structure hierarchy and the 2-STAR information (headlines on levels one to four) taken as input.

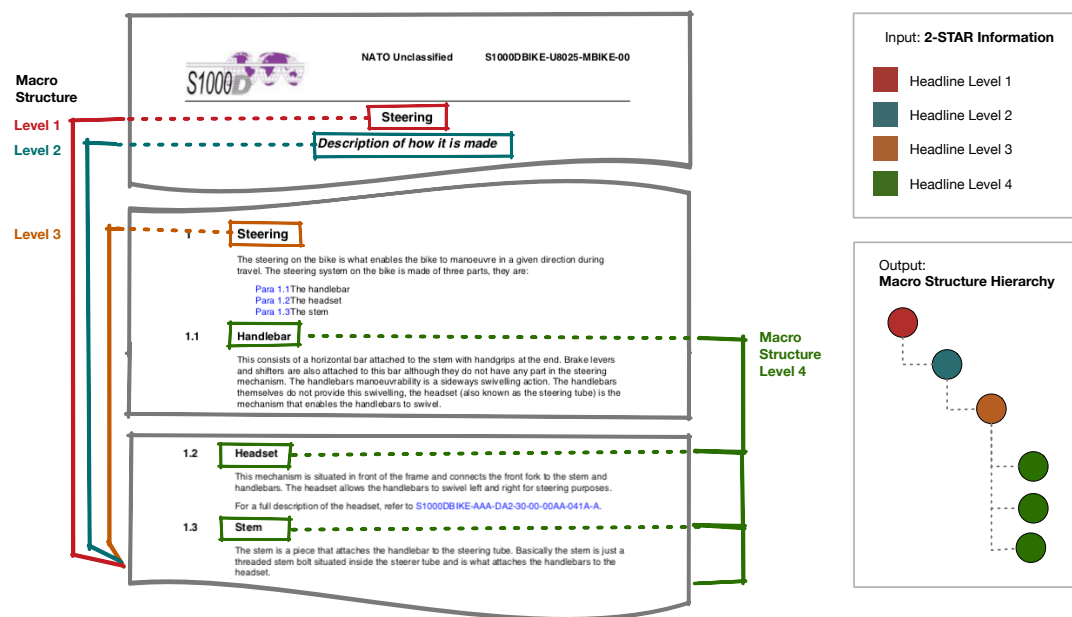


FIGURE 6.6 | Hands On: 3-STAR results from 2-STAR input.

The created macro structure instances are finally exported using CAPLAN's RDF export functionality in a 3-STAR Turtle RDF format (cf. Section 4.5.6). The following list shows a human-readable serialization of the recovered macro structure.

- Steering
  - Description of how it is made
    - \* Steering
      - Handlebar

- Headset
- Stem

Listings 6.9 – 6.14 show the resulting macro structure in the exported 3-STAR Turtle RDF [15] format. For each macro structure a `:InformationUnit` instance is defined. A hierarchy with parent and child relations is assembled using `:hasParent` and `:parentOf` statements respectively. Additionally, a title, a mime type, and the path to the file resource is defined. The path includes page and area (vertical position) information.

```
:Segment_TheBIKE2_3Manual_EN_0001_infount a :InformationUnit ;
  :parentOf :Segment_TheBIKE2_3Manual_EN_0002_infount ;
  :hasMimeType "application/pdf" ;
  :hasResource "TheBIKE2.3Manual_EN.pdf#page=1.11-2"@en ;
  :hasTitle "Steering"@en .
```

LISTING 6.9 | Hands On: Macro Structure in RDF (1/6).

```
:Segment_TheBIKE2_3Manual_EN_0002_infount a :InformationUnit ;
  :hasParent :Segment_TheBIKE2_3Manual_EN_0001_infount ;
  :parentOf :Segment_TheBIKE2_3Manual_EN_0003_infount ;
  :hasMimeType "application/pdf" ;
  :hasResource "TheBIKE2.3Manual_EN.pdf#page=1.14-2"@en ;
  :hasTitle "Description of how it is made"@en .
```

LISTING 6.10 | Hands On: Macro Structure in RDF (2/6).

```
:Segment_TheBIKE2_3Manual_EN_0003_infount a :InformationUnit ;
  :hasParent :Segment_TheBIKE2_3Manual_EN_0002_infount ;
  :parentOf
    :Segment_TheBIKE2_3Manual_EN_0004_infount ,
    :Segment_TheBIKE2_3Manual_EN_0005_infount ,
    :Segment_TheBIKE2_3Manual_EN_0006_infount ;
  :hasMimeType "application/pdf" ;
  :hasResource "TheBIKE2.3Manual_EN.pdf#page=1.63-2"@en ;
  :hasTitle "Steering"@en .
```

LISTING 6.11 | Hands On: Macro Structure in RDF (3/6).

```
:Segment_TheBIKE2_3Manual_EN_0004_infount a :InformationUnit ;
  :hasParent :Segment_TheBIKE2_3Manual_EN_0003_infount ;
  :hasMimeType "application/pdf" ;
  :hasResource "TheBIKE2.3Manual_EN.pdf#page=1.76-2.11"@en ;
  :hasTitle "Handlebar"@en .
```

LISTING 6.12 | Hands On: Macro Structure in RDF (4/6).

```
:Segment_TheBIKE2_3Manual_EN_0005_infount a :InformationUnit ;
  :hasParent :Segment_TheBIKE2_3Manual_EN_0003_infount ;
  :hasMimeType "application/pdf" ;
  :hasResource "TheBIKE2.3Manual_EN.pdf#page=2.11-2.20"@en ;
  :hasTitle "Headset"@en .
```

LISTING 6.13 | Hands On: Macro Structure in RDF (5/6).

```
:Segment_TheBIKE2_3Manual_EN_0006_infount a :InformationUnit ;
  :hasParent :Segment_TheBIKE2_3Manual_EN_0003_infount ;
  :hasMimeType "application/pdf" ;
  :hasResource "TheBIKE2.3Manual_EN.pdf#page=2.20-2"@en ;
  :hasTitle "Stem"@en .
```

LISTING 6.14 | Hands On: Macro Structure in RDF (6/6).

## 6.5 4-STAR: Information Typing

The 3-STAR semantification yielded a semantic representation of a macro structure hierarchy. The hierarchy consists of information units that represent chapters, sections and subsections. These semantically represented information units can also be used in semantic systems. However, they are not yet linked to concepts from other (external) ontologies. Thus, semantic information retrieval or other semantic services can not yet be realized.

Therefore, the 4-STAR semantification aims on linking information units to information types. Rhetorical filtering of information units is a typical use case example that becomes possible through this linking activity. Section 3.3.2 introduced a commonly agreed upon set of information types. Section 4.6.3 described the linking of information units to these information types using Automatic Document Classification approaches. These Automatic Document Classification require a decent amount of training data. Thus, Section 4.6.4 discussed practical alternatives to Automatic Document Classification in cases where training data is not available.

The S1000D Bike data set is rather small (cf. Section 7.2.3). Thus, Automatic Document Classification is not applicable. Instead, the practical alternative of exploiting patterns in the macro structure hierarchy was applied. In the S1000D Bike manual, each section (macro structure) on level 2 belongs to a fixed set of so called *Information Codes*. These information codes are defined in the S1000D specification and have a standardized verbalization. Table 6.1 shows a mapping of S1000D information codes used in the S1000D Bike manual to the common set of recommended information types (cf. Section 3.3.2).

Headline 2 / S1000D Information Code	Information Type
Description of how it is made	Description
Description of function	Operation
Description attributed to crew	Operation
Post-operation procedures (crew)	Operation
Other procedures to clean	Maintenance
Place on test stand	Test
Standard repair procedures	Repair
Remove and install a new item	Repair
Fill with air	Maintenance
Check pressure	Test
Fault reports and isolation procedures	Fault Isolation
Detected Fault	Fault Isolation
Remove procedures	Repair
Manual test	Test
Clean with rubbing alcohol	Maintenance
Install procedures	Repair
Oil	Maintenance
Clean with chain cleaning fluid	Maintenance
Clean with degreasing agent	Maintenance

TABLE 6.1 | Hands On: Information Type Mapping.

The information codes can be easily extracted from the S1000D Bike manual as the encapsulating micro structures (headlines on level 2) were recovered in the



2-STAR semantification step and are now directly accessible. The actual process of linking information units to the above information types is then realized as follows:

1. Iterate all macro structures / sections on level 2.
2. Extract all micro structures of type “Heading 2” from each macro structure
3. Extract the S1000D information code from the extracted micro structures.
4. Map the extracted S1000D information code to a common information type.
5. Link the macro structure to this information type.
6. Link all children / successors of the macro structure to this information type.
7. Link all parent / ancestors of the macro structure to this information type.

Figure 6.7 illustrates the described 4-STAR information typing.

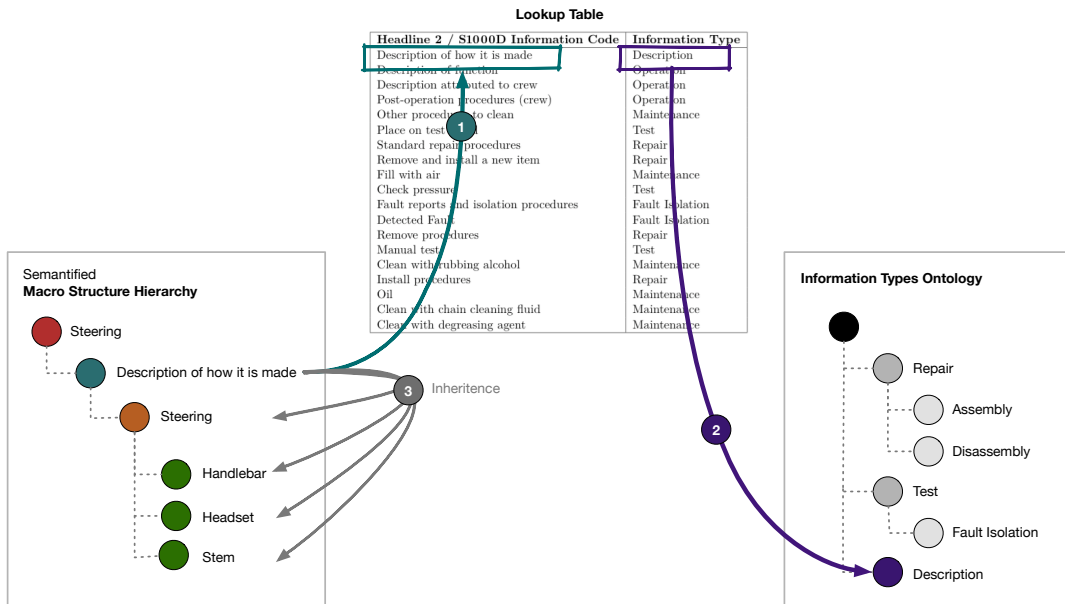


FIGURE 6.7 | Hands On: 4-STAR information typing.

The process has been implemented as a CAPLAN processor, which adds the respective links to the already available macro structure (information unit) instances. Then, again APOSTL’s integrated RDF export functionality is used to serialize the results in a 4-STAR Turtle RDF format. In contrast to the exported 3-STAR information, the resulting Turtle snippets (see Listing 6.15 – 6.20) contain an additional `hasInfoType` statement. This statement refers to the information type “Description” which was derived from the S1000D information code “Description of how it is made” found in the section that is titled respectively.

```

:Segment_TheBIKE2_3Manual_EN_0001_infounit a :InformationUnit ;
:parentOf :Segment_TheBIKE2_3Manual_EN_0002_infounit ;
:hasMimeType "application/pdf" ;
:hasResource "TheBIKE2.3Manual_EN.pdf#page=1.11-2"@en ;
:hasTitle "Steering"@en ;
:hasInfoType :Description .

```

LISTING 6.15 | Hands On: Information Types in RDF (1/6).

```

:Segment_TheBIKE2_3Manual_EN_0002_infounit a :InformationUnit ;
  :hasParent :Segment_TheBIKE2_3Manual_EN_0001_infounit ;
  :parentOf :Segment_TheBIKE2_3Manual_EN_0003_infounit ;
  :hasMimeType "application/pdf" ;
  :hasResource "TheBIKE2.3Manual_EN.pdf#page=1.14-2"@en ;
  :hasTitle "Description of how it is made"@en ;
  :hasInfoType :Description .

```

LISTING 6.16 | Hands On: Information Types in RDF (2/6).

```

:Segment_TheBIKE2_3Manual_EN_0003_infounit a :InformationUnit ;
  :hasParent :Segment_TheBIKE2_3Manual_EN_0002_infounit ;
  :parentOf
    :Segment_TheBIKE2_3Manual_EN_0004_infounit ,
    :Segment_TheBIKE2_3Manual_EN_0005_infounit ,
    :Segment_TheBIKE2_3Manual_EN_0006_infounit ;
  :hasMimeType "application/pdf" ;
  :hasResource "TheBIKE2.3Manual_EN.pdf#page=1.63-2"@en ;
  :hasTitle "Steering"@en ;
  :hasInfoType :Description .

```

LISTING 6.17 | Hands On: Information Types in RDF (3/6).

```

:Segment_TheBIKE2_3Manual_EN_0004_infounit a :InformationUnit ;
  :hasParent :Segment_TheBIKE2_3Manual_EN_0003_infounit ;
  :hasMimeType "application/pdf" ;
  :hasResource "TheBIKE2.3Manual_EN.pdf#page=1.76-2.11"@en ;
  :hasTitle "Handlebar"@en ;
  :hasInfoType :Description .

```

LISTING 6.18 | Hands On: Information Types in RDF (4/6).

```

:Segment_TheBIKE2_3Manual_EN_0005_infounit a :InformationUnit ;
  :hasParent :Segment_TheBIKE2_3Manual_EN_0003_infounit ;
  :hasMimeType "application/pdf" ;
  :hasResource "TheBIKE2.3Manual_EN.pdf#page=2.11-2.20"@en ;
  :hasTitle "Headset"@en ;
  :hasInfoType :Description .

```

LISTING 6.19 | Hands On: Information Types in RDF (5/6).

```

:Segment_TheBIKE2_3Manual_EN_0006_infounit a :InformationUnit ;
  :hasParent :Segment_TheBIKE2_3Manual_EN_0003_infounit ;
  :hasMimeType "application/pdf" ;
  :hasResource "TheBIKE2.3Manual_EN.pdf#page=2.20-2"@en ;
  :hasTitle "Stem"@en ;
  :hasInfoType :Description .

```

LISTING 6.20 | Hands On: Information Types in RDF (6/6).

## 6.6 5-STAR: Semantic Annotation

The 4-STAR semantification described in the previous section linked single macro structures (information units) to information type concepts. This was the first step towards a linked technical documentation and already enables the realization of improved semantic retrieval functionalities like rhetorical filtering. However, the final goal of linking the technical documents in a way that allows the access in a problem-oriented way has not yet achieved. Therefore, the information units need to be linked to concepts from an ontology that describes the corresponding machine.

### 6.6.1 Bike Ontology

The following sections describe the final semantification step of the S1000D bike documentation. In order to achieve a 5-STAR rating for the technical documents, the manual needs to be linked to an ontology that describes a bike. For this example, it is considered that such an ontology is externally provided.

Listing 6.21 shows a dash tree in KnowWE [10] markup that reflects the contents of an ontology describing a bike. Basically, the ontology models the physical breakdown of a bike, i.e., a hierarchy of components. It can be assumed that parent and child elements in the dash tree are compiled to `:subComponentOf` and `:hasSubComponent` relations respectively. The upper case strings in brackets are used to define the type of the respective elements and are compiled to `rdf:type` statements. For instance, the last four lines of the markup snippet define that the machine “Bike” has a sub-component called “Gears”, which itself has sub-components called “Mechs”, “Hubs” and “Shifters”.

```
%%TSM
Bike [MACHINE]
- Wheel [COMPONENT]
-- Inner Tube [COMPONENT]
-- Rear Wheel [COMPONENT]
-- Front Wheel [COMPONENT]
- Brake System [COMPONENT]
-- Brake Pads [COMPONENT]
- Steering [COMPONENT]
-- Stem [COMPONENT]
-- Headset [COMPONENT]
-- Handlebar [COMPONENT]
- Frame [COMPONENT]
-- Horn [COMPONENT]
- Drivetrain [COMPONENT]
-- Chain [COMPONENT]
- Gears [COMPONENT]
-- Mechs [COMPONENT]
-- Hubs [COMPONENT]
-- Shifters [COMPONENT]
%
```

LISTING 6.21 | Hands On: S1000D Bike Ontology.

### 6.6.2 Entity Recognition

The linking of documents to concepts from an ontology is primarily based on term matches occurring in the respective document texts. The set of terms is derived from the ontology, where each term refers to a concept label. A concept can have multiple labels. Having a set of terms available, the first task in the 5-STAR semantification step is finding all occurrences of terms in the texts of the involved documents. Therefore, an Entity Recognition approach based on fuzzy string matching techniques is employed (cf. Section 4.7.2). The Entity Recognition approach is implemented as an APOSTL processor. Table 6.2 shows the recognized entities for the following text of an already recovered information unit titled “Handlebar”.

Handlebar

This consists of a horizontal bar attached to the stem with handgrips at the end. Brake levers and shifters are also attached to this bar although

they do not have any part in the steering mechanism. The handlebars manoeuvrability is a sideways swivelling action. The handlebars themselves do not provide this swivelling, the headset (also known as the steering tube) is the mechanism that enables the handlebars to swivel.

Entity	Matched Text	Confidence
handlebar	Handlebar	1.0000
stem	stem	1.0000
shifters	shifters	1.0000
steering	steering	1.0000
mechs	mechanism	0.5333
handlebar	handlebars	0.8100
handlebar	handlebars	0.8100
headset	headset	1.0000
steering	steering	1.0000
mechs	mechanism	0.5333
handlebar	handlebars	0.8100

TABLE 6.2 | Hands On: Recognized Entities with Confidences.

### 6.6.3 Probabilistic Subject Analysis

The Entity Recognition yields a set of term matches, i.e., references to entities accompanied by the matched text in the document and an associated confidence that expresses the quality of the term match. Now, the task is to determine the most relevant subjects on basis of these term matches. Therefore, a Probabilistic Subject Analysis approach (cf. Section 4.7.4) is employed. This approach uses the following probabilistic model to first determine a probability for a term match representing a specific topic:

$$\mathbf{P}(\mathbf{topic} \mid \mathbf{match}) = \alpha * P(\mathit{topic} \mid \mathit{concept}) * P(\mathit{concept} \mid \mathit{term}) * P(\mathit{term} \mid \mathit{match}).$$

Therefore, it considers the quality of the term match  $P(\mathit{term} \mid \mathit{match})$ , the specificity of the term  $P(\mathit{concept} \mid \mathit{term})$ , and its relevance for a specific topic  $P(\mathit{topic} \mid \mathit{concept})$ . The quality of the term match  $P(\mathit{term} \mid \mathit{match})$  expresses the certainty that a given term match  $\mathit{match}$  actually represents a term  $\mathit{term}$  and thus corresponds to the confidence of the preceding Entity Recognition step (see Table 6.2).

The specificity  $P(\mathit{concept} \mid \mathit{term})$  expresses how specific a  $\mathit{term}$  is within the underlying ontology, i.e., how many  $\mathit{concepts}$  in the ontology share the same label. For instance, the  $\mathit{term}$  “brake pad” is the label of two  $\mathit{concepts}$  in the ontology, i.e., concepts representing the brake pads on the front wheel and the back wheel. The respective information can be derived from the ontology using SPARQL queries counting the number of concepts with the same label and a function transforming the count into a probability. Different functions can be employed to transform the concept count into a probability. In this example the following simple specificity function is used:

$$\mathbf{P}(\mathbf{concept} \mid \mathbf{term}) = \frac{1}{\mathit{frequency}}.$$

This function equally distributes the probability over all concepts, yielding a specificity of 0.5 for the term “brake pad” and the two associated concepts.

The relevance  $P(\text{topic} \mid \text{concept})$  expresses how relevant a *concept* is for a certain *topic*. While *topic* also refers to a concept from the ontology it is not necessarily equal to the *concept* parameter. Instead, the relevance considers ontological information to determine the relevance between two concepts. For instance, consider the ontology described in Listing 6.21 and Table 6.3 for the derived relevances for the *concept* “Stem” and related *topics* “Stem”, “Steering”, “Handlebar” and “Headset”.

Topic	Concept	Relevance	Explanation
Stem	Stem	1,0000	Identity
Stem	Steering	0,7508	:subComponentOf
Stem	Handlebar	0,3546	:subComponentOf/:hasSubComponent (sibling)
Stem	Headset	0,3546	:subComponentOf/:hasSubComponent (sibling)

TABLE 6.3 | Hands On: Recognized Entities with Confidences.

The table shows that the concept “Stem” is most relevant for the topic representing itself. Then, it is also relevant for the parent component in the bike (the concept “Stem” is modeled as sub-component of the component “Steering”). Additionally, it is also considered relevant for concepts that describe sibling concepts like “Handlebar” and “Headset”, which both are also sub-components of the “Steering” component.

Relevances as listed in Table 6.3 can again be derived from the ontology using SPARQL queries that count the (weighted) distance between concepts and a function that transforms the distance into a probability. The distance calculation might weight distances differently depending on the relation that connects two concepts, e.g. a sub-component relation might be considered more important than a relation that connects functions and components. In the example, the following function was employed to transform the weighted distances to probabilities:

$$\mathbf{P}(\text{topic} \mid \text{concept}) = \frac{1 + e^{-1.5 \cdot 1.5}}{1 + e^{(\text{distance} - 1.5) \cdot 1.5}} .$$

Using concrete values for term quality  $P(\text{term} \mid \text{match})$ , specificity  $P(\text{concept} \mid \text{term})$ , and relevance  $P(\text{topic} \mid \text{concept})$  the probability for a topic given a term match  $P(\text{topic} \mid \text{match})$  is computed for each term match. For instance, consider one of the term matches “handlebar” listed in Table 6.2 for the information unit “Handlebar” and the topic “Stem”. The term quality (`'handlebar'|'handlebars'`) is defined as 0.81. The specificity of the term “handlebar” is distributed over one single concept “Handlebar” and is thus 1.0. The relevance  $P(\text{Stem} \mid \text{Handlebar})$  is defined as 0.3546. Thus, the probability for the topic “Stem” given a term match “handlebars” is:

$$\mathbf{P}(\text{Stem} \mid \text{'handlebar'}) = \alpha * 0.3546 * 1.0 * 0.81 .$$

The parameter  $\alpha$  reflects the linguistic uncertainty and can either be a fixed or an adaptive value. A dynamic value might consider the type of the micro structure a term match occurred in to reduce the linguistic uncertainty. In this example the parameter  $\alpha$  was an adaptive value that considered the covering micro structure of a term match:

$$\alpha = 1 - 0.33^{\text{weight}}$$

The internal parameter *weight* was set to 4.0 (resulting in 0.9881) for all types of headings and 1.0 (resulting in 0.67) for all other types of micro structures.

This way, the probabilities for all topics given certain term matches  $P(\text{topic} \mid \text{match})$  are computed. Finally, an overall probability for each topic is computed (cf.

Section 4.7.4):

$$\mathbf{P}(\mathbf{topic}) = 1 - \prod_{\mathit{match}}^{M_i} (1 - P(\mathit{topic} \mid \mathit{match})) .$$

Figure 6.8 summarizes the 5-STAR semantic annotation step. This process does not necessarily determine one single concept as topic for an information unit but lists all concepts together with the computed probabilities that are considered highly relevant.

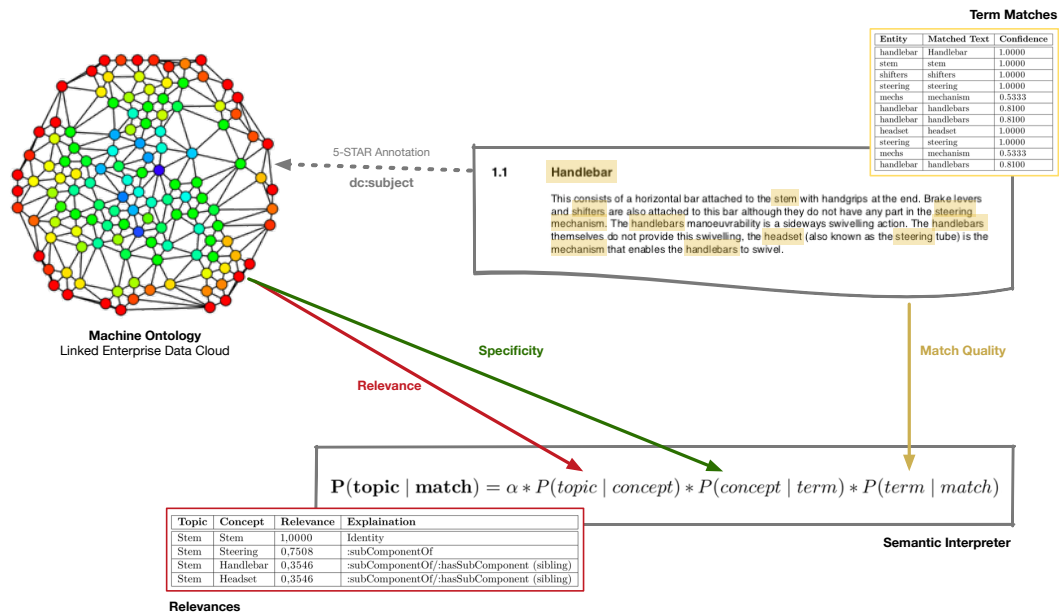


FIGURE 6.8 | Hands On: 5-STAR semantic annotation.

The resulting probabilities can be exploited from consuming applications like Semantic Search. For instance, a semantic search engine could exploit the probability while looking up and ranking search results (information units) according to its inference result. The most important topic probabilities for the aforementioned information unit “Handlebar” are listed in Table 6.4.

Topic	Probability
Handlebar	0.9995
Steering	0.9816
Headset	0.9478
Stem	0.9478

TABLE 6.4 | Hands On: Topic Probabilities.

### 6.6.4 Output

The complete Subject Analysis task has been implemented as a CAPLAN processor which adds the respective topic links to the already available macro structure / information unit instances. The domain-specific background knowledge for relevance and linguistic uncertainty can be defined by the user. APOSTL’s integrated RDF export functionality was used to serialize the results in a 5-STAR Turtle RDF format.

In contrast to the exported 4-STAR information, the resulting Turtle snippets (see Listing 6.22 – 6.25) contain additional `hasAnnotation` statements that refers to the concepts and the computed probabilities.

```
:Segment_TheBIKE2_3Manual_EN_0003_infounit a :InformationUnit ;
:hasParent :Segment_TheBIKE2_3Manual_EN_0002_infounit ;
:parentOf
:Segment_TheBIKE2_3Manual_EN_0004_infounit ,
:Segment_TheBIKE2_3Manual_EN_0005_infounit ,
:Segment_TheBIKE2_3Manual_EN_0006_infounit ;
:hasMimeType "application/pdf" ;
:hasResource "TheBIKE2.3Manual_EN.pdf#page=1.63-2"@en ;
:hasTitle "Steering"@en ;
:hasInfoType :Description ;
:hasAnnotation [ :hasScore "0.9997" ; :hasConcept :Steering ] ;
:hasAnnotation [ :hasScore "0.9932" ; :hasConcept :Handlebar ] ;
:hasAnnotation [ :hasScore "0.9916" ; :hasConcept :Stem ] ;
:hasAnnotation [ :hasScore "0.9892" ; :hasConcept :Headset ] .
```

LISTING 6.22 | Hands On: Semantic Annotations in RDF (3/6).

```
:Segment_TheBIKE2_3Manual_EN_0004_infounit a :InformationUnit ;
:hasParent :Segment_TheBIKE2_3Manual_EN_0003_infounit ;
:hasMimeType "application/pdf" ;
:hasResource "TheBIKE2.3Manual_EN.pdf#page=1.76-2.11"@en ;
:hasTitle "Handlebar"@en ;
:hasInfoType :Description ;
:hasAnnotation [ :hasScore "0.9995" ; :hasConcept :Handlebar ] ;
:hasAnnotation [ :hasScore "0.9816" ; :hasConcept :Steering ] ;
:hasAnnotation [ :hasScore "0.9478" ; :hasConcept :Headset ] ;
:hasAnnotation [ :hasScore "0.9478" ; :hasConcept :Stem ] .
```

LISTING 6.23 | Hands On: Semantic Annotations in RDF (4/6).

```
:Segment_TheBIKE2_3Manual_EN_0005_infounit a :InformationUnit ;
:hasParent :Segment_TheBIKE2_3Manual_EN_0003_infounit ;
:hasMimeType "application/pdf" ;
:hasResource "TheBIKE2.3Manual_EN.pdf#page=2.11-2.20"@en ;
:hasTitle "Headset"@en ;
:hasInfoType :Description ;
:hasAnnotation [ :hasScore "0.9990" ; :hasConcept :Headset ] ;
:hasAnnotation [ :hasScore "0.9778" ; :hasConcept :Steering ] ;
:hasAnnotation [ :hasScore "0.9379" ; :hasConcept :Stem ] .
```

LISTING 6.24 | Hands On: Semantic Annotations in RDF (5/6).

```
:Segment_TheBIKE2_3Manual_EN_0006_infounit a :InformationUnit ;
:hasParent :Segment_TheBIKE2_3Manual_EN_0003_infounit ;
:hasMimeType "application/pdf" ;
:hasResource "TheBIKE2.3Manual_EN.pdf#page=2.20-2"@en ;
:hasTitle "Stem"@en ;
:hasInfoType :Description ;
:hasAnnotation [ :hasScore "0.9993" ; :hasConcept :Stem ] ;
:hasAnnotation [ :hasScore "0.9886" ; :hasConcept :Steering ] ;
:hasAnnotation [ :hasScore "0.9789" ; :hasConcept :Handlebar ] ;
:hasAnnotation [ :hasScore "0.9628" ; :hasConcept :Headset ] .
```

LISTING 6.25 | Hands On: Semantic Annotations in RDF (6/6).

## 6.7 A Demonstration of Improved Accessibility

The previous sections demonstrated the 5-STAR semantification process using the “Steering” chapter of the S1000D Bike manual. The chapter contains one section which itself has three subsections. The resulting macro structure has first been transformed into information units. Then, they were subsequently annotated with information types and components from an external ontology. The resulting 5-STAR information is available in the standardized Turtle RDF [15] format. State-of-the-art semantic information systems can exploit the added information in order to improve the retrieval and presentation of technical documents. Figure 6.9 shows the information unit “Headset” from the semantified “Steering” chapter. The 3-STAR macro structure information is exploited to limit the document presentation to the relevant part that represents the respective section. Preceding this improved information presentation a semantic search for descriptive information (information type :Description) and the concept :Headset was executed. In summary, the demonstrated 5-STAR semantification process achieved its target of improving the accessibility of the S1000D Bike manual. Unlike the raw PDF document that was used as input, technical information about the S1000D Bike is now accessible in a problem-oriented and targeted manner.

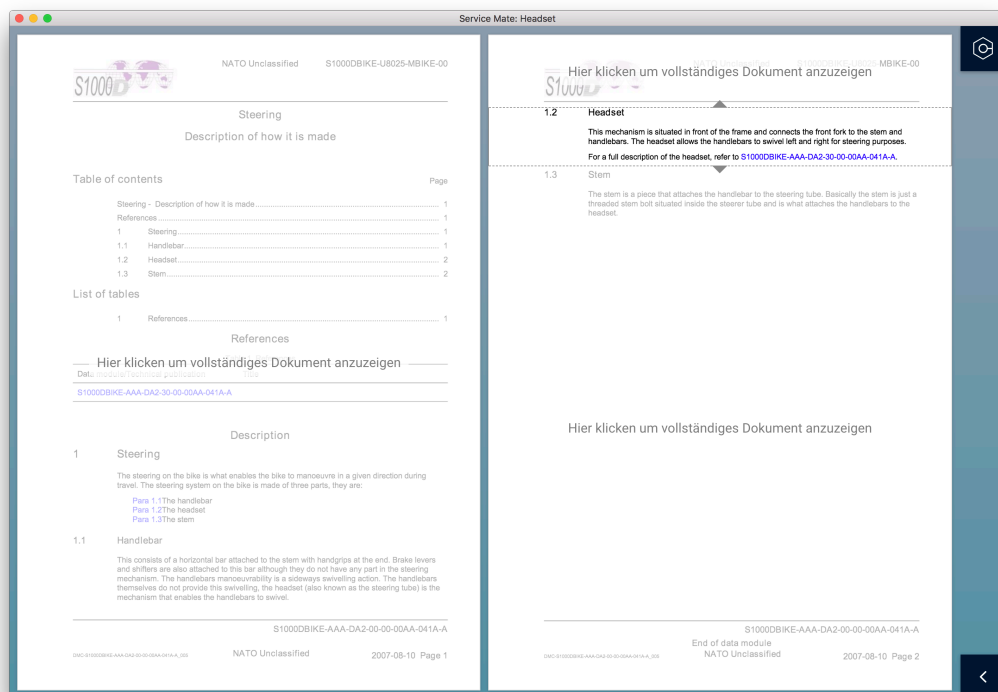


FIGURE 6.9 | Hands On: Improved Accessibility in Service Mate.



## Chapter 7

# Experiences

---

*The only source of knowledge is  
experience.*

---

*Albert Einstein*

## 7.1 Overview

This chapter describes a series of case studies of semantification projects for technical documentation where several of the presented approaches has been employed. The semantification projects have quite heterogeneous settings considering the underlying machinery, the size of the corpus and the required semantification steps. For each project a brief introduction of the scenario and a short summary of the semantification process is given.

## 7.2 S1000D Bike

The S1000D Bike is an evaluation data set consisting of technical publications for a fictional bike. The data set is designed to enable the evaluation of wide-spread S1000D tools.

### 7.2.1 Introduction

S1000D is an information model for technical documentation that is widely used in the aerospace industry and military organizations. The S1000D information model has been introduced in detail in Section 3.4.5. Basically, a S1000D publication consists of data modules which describe certain topics of interest. Each data module is self contained and encoded with a data module code. The source format of S1000D publications is SGML/XML. However, data modules can be compiled to complete publications in various formats like PDF or HTML. Publications available in S1000D source format usually receive four to five stars in the 5-STAR maturity schema. However, the available data set is a perfect source for the evaluation of the presented semantification process as the data set is available in S1000D XML source format and as compiled PDF publication.

### 7.2.2 Goals and Application Scenarios

Technical documentation written according to the S1000D information model is already modularized. Data modules contain the modularized content and are encoded with a data module code. The data module code allows to derive the information type and the subject/topic of the module. Thus, the primary application scenario of technical documents in S1000D format is the targeted information supply of people working with the documents.

#### **Goal 1: Ontology Derivation from Core Documentation Entities**

The 5-STAR semantification requires on its final stage an ontology that represents the underlying real-world entities in order to be able to annotate the modularized content with respective concepts. An ontology describing the S1000D Bike, however, is not available. As Core Documentation Entities can directly be derived from S1000D XML source files, the first goal of this case study is to derive a technical ontology. The

derived ontology contains concepts that represent components of the S1000D Bike. The automatically generated ontology will be evaluated manually for reasonability (see Section 7.2.6).

## **Goal 2: Semantification of PDF documents**

The main purpose of this case study, however, is the semantification of the compiled PDF documentation, which is not modularized and not semantically annotated. Hence, the second goal of the case study is to produce a PDF-based data set that is compatible with a state-of-the-art semantic information system (see Section 7.2.7). The data set based on PDF resources, therefore, gets semantically prepared equally to the original S1000D XML sources, i.e. modularized and semantically annotated. Therefore, the PDF documents are subsequently semantified by running through respective semantification steps. For the fifth step in the semantification process the extracted ontology (see Goal 1) is used. The semantified PDF-based data set is evaluated against the S1000D XML source files (see Section 7.2.6).

### **7.2.3 Data Set Description**

The S1000D Bike data set accompanies the official S1000D specification and is intended for evaluation purposes. The data set comprises a PDF document representing a technical manual for a fictional bicycle. The PDF document has 188 pages and has been compiled from S1000D source files. The underlying source files are distributed over 41 illustrations and 57 data modules in both SGML and XML format.

### **7.2.4 The Semantification Process**

This section describes the semantification architecture used to semantify the described data set. Section 7.2.4 describes the two-fold semantification architecture. Section 7.2.5 describes the involved knowledge resources.

#### **Semantification Architecture**

The semantification architecture is spread over two different pipelines. A first pipeline aims on extracting an ontology from S1000D XML files. A second pipeline is applied for the complete semantification of the corresponding 1-STAR PDF file, i.e. the transformation into a modularized and semantically prepared 5-STAR documentation.

#### **S1000D Semantification Architecture**

The first pipeline (see Figure 7.1) initially reads the S1000D XML source files. The S1000D source files initially have a 3-STAR rating. Subsequent processing steps discover Core Documentation Entities in the source files and add metadata in forms of information types and concept annotations to the S1000D XML source files. Then, the processed S1000D XML files yield a 5-STAR rating. Although this semantification process is not required in order to perform the semantification of PDF files it is the basis for the subsequent evaluation. Additionally, the semantification enables the derivation of an ontology from the S1000D source files (see Goal 2 described in Section 7.2.2).

The exploitation of the data module code (DMC) of S1000D XML files gives access to the information type (B). The Core Documentation Entity discovery (C) exploits structural components in S1000D XML while considering the information type of the data module in focus. The ontology export (D) mechanism considers

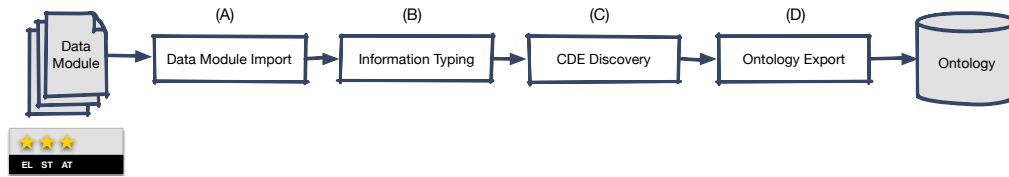


FIGURE 7.1 | S1000D: Data Module Semantification Architecture.

all concepts that have been derived from the Data Module Code for the annotation process and additionally puts them in an hierarchical order according to the Standard Numbering System that is part of the S1000D specification.

### PDF Semantification Architecture

The second pipeline (see Figure 7.2) initially reads the PDF file (1-STAR rating). Subsequent processing steps aim on recovering nano (A), micro (B), and macro (C) structures. These recovered structures already allow to split the PDF to smaller modules. Finally, a semantic annotation process (D) determines concepts from an external ontology for the semantic annotation of the PDF modules. These PDF modules finally yield 5-STARS in the maturity schema.

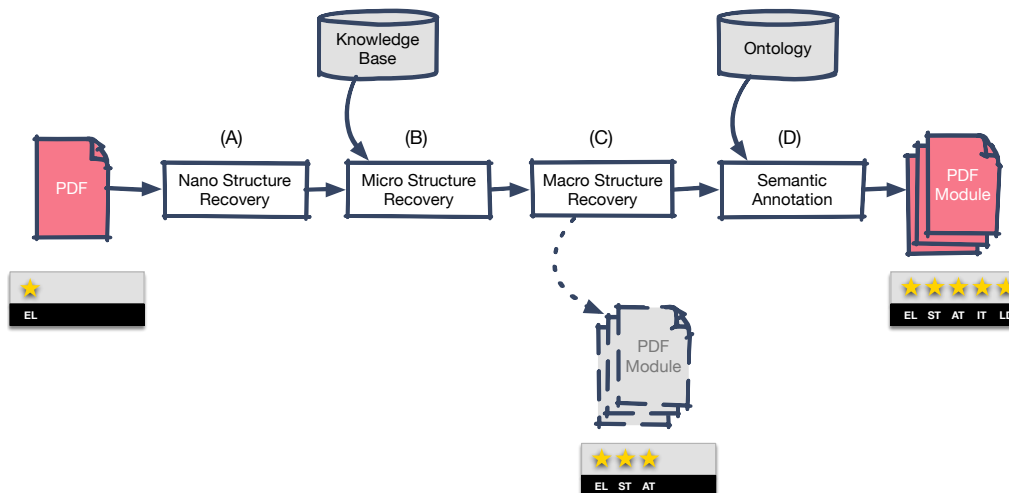


FIGURE 7.2 | S1000D: PDF Semantification Architecture.

The nano structure recovery (A) is based on the open source tool pdf2xml that has been proposed by Dejean et al. [52]. The micro structure recovery (B) uses the knowledge base described in Section 7.2.5 to estimate micro structures. The macro structure recovery (C) works upon the recovered micro structures and employs the macro structure recovery mechanisms described in Section 4.5.3. The semantic annotation (D) is based on Probabilistic Explicit Semantic Analysis (PESA) described in Section 4.7.4 and uses the extracted ontology.

The probabilistic model has been parametrized to incorporate existing domain knowledge. Therefore, the tailoring possibilities described in Section 4.7.4 were used as follows:

- **Term Confidence**  $P(t | match)$ : Order independent lookups without decreasing term confidences were allowed. Matches that had only been possible due to stemming were discriminated.

- **Term Specificity**  $P(c | t)$ : The specificity was distributed equally over all concepts, i.e. if a term is attached to two concepts, the specificity of the term is 0.5 for both concepts.
- **Concept Relevance**  $P(topic | c)$ : For operating manuals the refer property was slightly preferred, in descriptive manuals the subComponentOf property respectively<sup>1</sup>.
- **Linguistic Uncertainty**  $\alpha_{adaptive}$ : Weights  $w$  greater than 1.0 were defined for headlines, i.e. term matches occurring in the heading of sections were preferred.

The respective semantic interpreter works upon 38 terms and 21 concepts using 888 semantic relations. Figure 7.3 shows an excerpt of the serialized semantic interpreter.

Term	Specificity => P(concept term)	Concept Label	Relevance => P(topic concept)	Concept URI
bicycle	1	Bicycle	1,0000	http://denkbare.com/ServiceMate/Bike/concept#AAA-000-00-00-00
bicycle	1	AAA-D05-00-00-00	0,7508	http://denkbare.com/ServiceMate/Bike/concept#AAA-D05-00-00-00
bicycle	1	Wheel	0,7508	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA0-00-00-00
bicycle	1	AAA-DA0-10-00-00	0,7508	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA0-10-00-00
bicycle	1	Brake system	0,7508	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA1-00-00-00
bicycle	1	Steering	0,7508	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA2-00-00-00
bicycle	1	Frame	0,7508	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA3-00-00-00
bicycle	1	Drivetrain	0,7508	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA4-00-00-00
bicycle	1	Gears	0,7508	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA5-00-00-00
bicycle	1	Bicycle	0,3546	http://denkbare.com/ServiceMate/Bike/concept#AAA-D05-10-00-00
bicycle	1	Bicycle	0,3546	http://denkbare.com/ServiceMate/Bike/concept#AAA-D05-20-00-00
bicycle	1	Bicycle	0,3546	http://denkbare.com/ServiceMate/Bike/concept#AAA-D05-40-00-00
bicycle	1	Inner tube	0,3546	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA0-10-10-00
bicycle	1	Front wheel	0,3546	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA0-10-20-00
bicycle	1	Rear wheel	0,3546	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA0-20-00-00
bicycle	1	Brake Pads	0,3546	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA1-10-00-00
bicycle	1	Stem	0,3546	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA2-10-00-00
bicycle	1	Handlebar	0,3546	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA2-20-00-00
bicycle	1	Headset	0,3546	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA2-30-00-00
bicycle	1	Horn	0,3546	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA3-10-00-00
bicycle	1	Chain	0,3546	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA4-10-00-00
bicycle	1	Mechs	0,3546	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA5-10-00-00
bicycle	1	Hubs	0,3546	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA5-20-00-00
bicycle	1	Shifters	0,3546	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA5-30-00-00
brake pads	1	Brake Pads	1,0000	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA1-10-00-00
brake pads	1	Brake system	0,7508	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA1-00-00-00
brake pads	1	Bicycle	0,3546	http://denkbare.com/ServiceMate/Bike/concept#AAA-000-00-00-00
brake pads	1	AAA-D05-00-00-00	0,1054	http://denkbare.com/ServiceMate/Bike/concept#AAA-D05-00-00-00
brake pads	1	Wheel	0,1054	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA0-00-00-00
brake pads	1	AAA-DA0-10-00-00	0,1054	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA0-10-00-00
brake pads	1	Steering	0,1054	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA2-00-00-00
brake pads	1	Frame	0,1054	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA3-00-00-00
brake pads	1	Drivetrain	0,1054	http://denkbare.com/ServiceMate/Bike/concept#AAA-DA4-00-00-00

FIGURE 7.3 | S1000D: Serialized Semantic Interpreter.

## 7.2.5 Knowledge Resources

For this case studies two kinds of knowledge resources are relevant. The following section first describes the S1000D Bike ontology. Afterwards, the knowledge base carrying the classification knowledge for the micro structure recovery is explained.

### S1000D Ontology

The ontology providing the concepts for the semantic annotation step has been automatically extracted from S1000D XML source data. The ontology contains information about the hierarchical structure of components in the S1000D Bike as well as

<sup>1</sup>The properties are explained as part of the ontology description in Section 7.2.5

functional connections between components (see Figure 7.4 for a simplified visualization). Labels are attached to all concepts. Figure 7.5 is an excerpt of the extracted ontology that represents the component hierarchy.

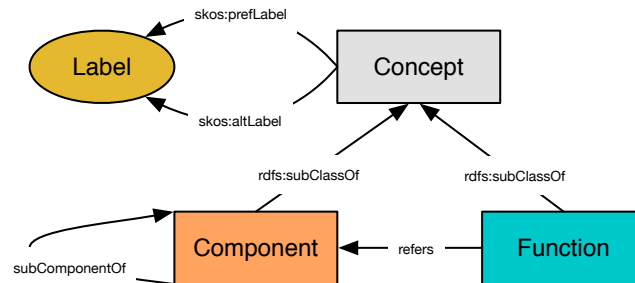


FIGURE 7.4 | S1000D: Structure of the extracted Ontology.

label	type	location	concept
▼ Wheel	Maschine		bike:concept#AAA-DA0-00-00-00
▼ -- (unspecific location)	Location		concept#AAA-DA0-00-00-00_unspecific_location
Rear wheel	Komponente		bike:concept#AAA-DA0-20-00-00
▼ Brake system	Maschine		bike:concept#AAA-DA1-00-00-00
▼ -- (unspecific location)	Location		concept#AAA-DA1-00-00-00_unspecific_location
Brake pads	Komponente		bike:concept#AAA-DA1-10-00-00
▼ Steering	Maschine		bike:concept#AAA-DA2-00-00-00
▼ -- (unspecific location)	Location		concept#AAA-DA2-00-00-00_unspecific_location
Stem	Komponente		bike:concept#AAA-DA2-10-00-00
Handlebar	Komponente		bike:concept#AAA-DA2-20-00-00
Headset	Komponente		bike:concept#AAA-DA2-30-00-00
▼ Frame	Maschine		bike:concept#AAA-DA3-00-00-00
▼ -- (unspecific location)	Location		concept#AAA-DA3-00-00-00_unspecific_location
Horn	Komponente		bike:concept#AAA-DA3-10-00-00
▼ Drivetrain	Maschine		bike:concept#AAA-DA4-00-00-00
▼ -- (unspecific location)	Location		concept#AAA-DA4-00-00-00_unspecific_location
Chain   Drive train	Komponente		bike:concept#AAA-DA4-10-00-00
▼ Gears	Maschine		bike:concept#AAA-DA5-00-00-00
▼ -- (unspecific location)	Location		concept#AAA-DA5-00-00-00_unspecific_location
Mechs	Komponente		bike:concept#AAA-DA5-10-00-00
Hubs	Komponente		bike:concept#AAA-DA5-20-00-00
Shifters	Komponente		bike:concept#AAA-DA5-30-00-00

FIGURE 7.5 | S1000D: Extracted Ontology.

## TEKNO Knowledge Base

The 2-STAR semantification requires a knowledge base that contains classification knowledge for micro structures. For this case study a respective knowledge base has been assembled using TEKNO Studio. The knowledge base contains eleven Set-Covering models for the following micro structure types:

- Title
- Heading 1
- Heading 2
- Heading 3
- Heading 4
- Table of Contents
- Procedure
- Caution
- Caution Text
- Warning
- Caption

Each of the Set-Covering models has on average 6.45 Set-Covering relations that reflect formatting features of the respective micro structure type. The Set-Covering model has first been created fully automatically using the model suggestion functionality of TEKNO Studio. After reviewing the results of the automatically generated model minor adaptations to the Set-Covering relations of one model have been made. Figure 7.6 shows an example of recovered micro structure in TEKNO Studio.

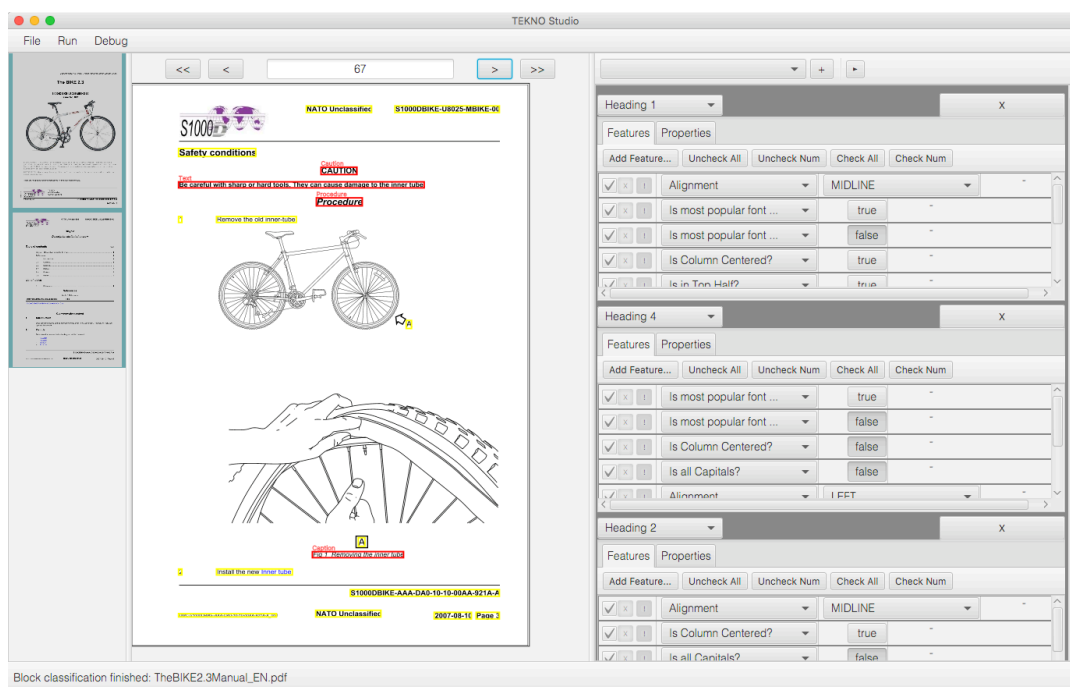


FIGURE 7.6 | S1000D: Set-Covering Model in TEKNO Studio.

### 7.2.6 Evaluation

This section describes the evaluation of the semantification architecture with respect to the S1000D Bike data set. First the employed evaluation architecture is presented. Then, the employed evaluation methods are described. Finally, evaluation results are listed .

#### Evaluation Architecture

For the evaluation of the semantification results an evaluation architecture has been implemented (see Figure 7.7). First, the XML data modules are aligned with the results of the PDF semantification pipeline. Therefore, an automatic alignment (A) of the respective S1000D XML and PDF modules is generated on basis of textual features of both data sources. Then, the results of the different semantification steps get evaluated separately by respective evaluation components (B – D). The evaluation mechanisms consider the S1000D XML modules as gold standard. The evaluation methods employed are described in detail in the next section.

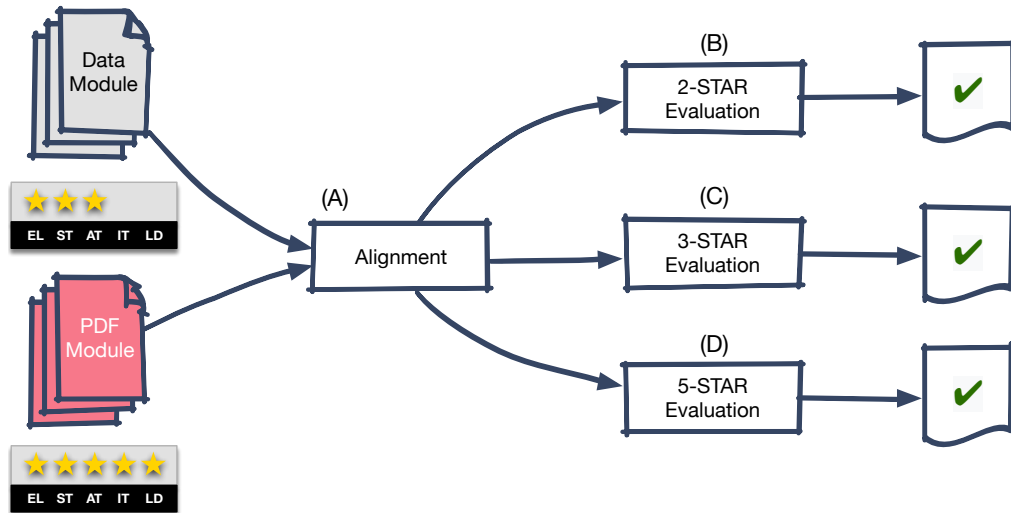


FIGURE 7.7 | S1000D Evaluation Architecture.

#### Evaluation Methods

The evaluation of the S1000D Bike data set comprises different semantification steps. The evaluation methods are slightly different for the respective results. All evaluations rely on a correct alignment of the S1000D XML data modules and the PDF modules.

The **2-STAR** semantification aims on the recovery of micro structures like headlines, paragraphs, or lists. The S1000D XML source files provide respective markup for these elements, i.e. they are directly accessible. The semantification architecture for the PDF file determines corresponding micro structures for the PDF modules. Having micro structures available in both the S1000D XML source files and the PDF modules, the accuracy of the micro structure recovery processor can be determined. The accuracy is defined as follows:

**Definition 7.2.1** (Micro Structure Recovery Accuracy). The micro structure recovery accuracy  $acc_{2-STAR}$  is defined as the number of correctly classified blocks  $M_{c,pdf} \subseteq M_{pdf}$  in the PDF modules with regard to all blocks  $M_{xml}$  available from



S1000D XML data modules:

$$acc_{2-STAR} = \frac{|M_{c, pdf}|}{|M_{xml}|}.$$

A block  $p \in M_{pdf}$  is correctly classified, if there is a block  $x \in M_{xml}$  with the same type and text assigned.

The **3-STAR** semantification aims on the recovery of macro structures. In the context of S1000D data modules this means cutting down the complete PDF to small self-contained and deduplicated modules. Having modules available as S1000D XML files and as PDF modules the macro structure recovery can be evaluated by comparing the text of aligned modules. Therefore, the macro structure recovery cost gets determined, which is defined as follows:

**Definition 7.2.2** (Macro Structure Recovery Cost). The macro structure recovery cost  $cost_{3-STAR}$  is defined as the minimum number of edit transformations that transforms the text of a macro structure  $a \in S_{pdf}$  to the text of an aligned macro structure  $b \in S_{xml}$ . The set of allowed edit operations comprises the *insertion* of a single character, the *deletion* of a single character and the *substitution* of a single character. This corresponds to the well established distance function proposed by Levenshtein [118].

The **5-STAR** semantification aims on annotating modules with concepts from an external ontology. The data module code that identifies a data module in S1000D XML provides information about the topic of the module, such that an annotation concept can be derived. The semantification process for the PDF document involves a semantic annotation step which assigns concept annotations to PDF modules. After the semantification, both S1000D XML data modules and PDF modules have concept annotations attached. The annotation of PDF modules was limited to exactly one annotation per module. Having exactly one annotation available for each S1000D XML source files and the corresponding PDF module, the accuracy of the semantic annotation can be determined. The 5-STAR accuracy is defined as follows:

**Definition 7.2.3** (Annotation Accuracy). The annotation accuracy  $acc_{5-STAR}$  is defined as the number of correct annotations  $A_{c, pdf} \subseteq A_{pdf}$  in the PDF modules with regard to all annotations  $A_{xml}$  available from S1000D XML data modules:

$$acc_{5-STAR} = \frac{|A_{c, pdf}|}{|A_{xml}|}.$$

An annotation  $a \in A_{pdf}$  is correct, if there is an annotation  $b \in A_{xml}$  with the same URI.

## Evaluation Results

This sections lists the results of the different evaluation methods. The results of the 2-STAR, 3-STAR, and 5-STAR evaluation are separated over the following paragraphs.

### 2-STAR Results

In total, 217 micro structures have been automatically classified using the assembled knowledge base. The classification using the automatically generated knowledge base has yielded the results listed in Table 7.1. The micro structures were classified with an average accuracy of 98%, i.e. 211 micro structures were classified correctly. The

six micro structures instances that were misclassified all belong to the class “Caption”. The primary reason for the classification error was a bad blockification (1-STAR Nano Structure Recovery).

Micro Structure	Occurrences	Correct	Accuracy
Title	1	1	100%
Heading 1	36	36	100%
Heading 2	36	36	100%
Heading 3	14	14	100%
Heading 4	16	16	100%
Table of Contents	31	31	100%
Procedure	24	24	100%
Caution	11	11	100%
Caution Text	7	7	100%
Warning	7	7	100%
Caption	34	28	82%
<b>Total</b>	<b>217</b>	<b>211</b>	<b>98%</b>

TABLE 7.1 | S1000D: Macro Structure Recovery Costs

### 3-STAR

Following the 5-STAR semantification idea, the classified micro structures have been used as the basis for the further processing. The 3-STAR semantification aimed on recovering macro structures (modules) from micro structures. The plain text of the recovered PDF-based modules has been compared to the text of the XML-based S1000D data modules. Table 7.2 lists the recovery cost for each module.

As the recovery costs of some modules are rather high the plain text of the modules have been inspected manually. The manual inspection showed that the recovered PDF modules contain automatically generated text that is not available in the XML-based S1000D source data, e.g., table of contents, references, list of tables, and figures at the beginning of each section. The inspection also showed that despite this additional text the modules had been correctly recovered.

Module Title	Recovery Cost
Bicycle - Description attributed to crew	207
Bicycle - Description of function	118
Bicycle - Description of how it is made	274
Bicycle - Other procedures to clean	1075
Bicycle - Place on test stand	2166
Bicycle - Standard repair procedures	2985
Brake pads - Clean with rubbing alcohol	2242
Brake system - Description of how it is made	509
Brake system - Manual test	2153
Chain - Clean with chain cleaning fluid	537
Chain - Oil	785
Drivetrain - Description of how it is made	1
Frame - Description of how it is made	144
Front wheel - Fault reports and isolation procedures	2329
Gears - Description of how it is made	28
Handlebar - Install procedures	971
Handlebar - Remove procedures	451
Headset - Description of how it is made	32
Headset - Install procedures	689
Headset - Remove procedures	629
Horn - Remove and install a new item	457
Hubs - Clean with degreasing agent	776
Inner tube - Remove and install a new item	630
Mechs - Description of how it is made	145
Rear wheel - Detected fault	0
Rear wheel - Remove procedures	144
Shifters - Description of how it is made	75
Steering - Description of how it is made	227
Stem - Install procedures	738
Stem - Remove procedures	656
Tire - Check pressure	2266
Tire - Fill with air	2236
Tire - Remove and install a new item	2390
Wheel - Description of how it is made	489

TABLE 7.2 | S1000D: Macro Structure Recovery Costs

### 5-STAR

An additional semantification step performed the Subject Analysis task on the recovered macro structures. Table 7.3 reports on the results in forms of expected and retrieved annotations, the confidence of the retrieved annotations and an indicator whether the annotation is correct.

Module Title	Expected	Retrieved	Confidence	Correct
Bicycle - Description attributed to crew	AAA-D00-00-00-00	AAA-000-00-00-00	0.9995	NO
Bicycle - Description of function	AAA-D00-00-00-00	AAA-000-00-00-00	0.9999	NO
Bicycle - Description of how it is made	AAA-D00-00-00-00	AAA-000-00-00-00	0.9996	NO
Bicycle - Other procedures to clean	AAA-D00-00-00-00	AAA-000-00-00-00	0.9990	NO
Bicycle - Place on test stand	AAA-D00-00-00-00	AAA-000-00-00-00	0.9960	NO
Bicycle - Standard repair procedures	AAA-D00-00-00-00	AAA-000-00-00-00	0.9974	NO
Brake pads - Clean with rubbing alcohol	AAA-DA1-10-00-00	AAA-DA1-10-00-00	0.9960	YES
Brake system - Description of how it is made	AAA-DA1-00-00-00	AAA-DA1-00-00-00	0.9997	YES
Brake system - Manual test	AAA-DA1-00-00-00	AAA-DA1-00-00-00	0.9984	YES
Chain - Clean with chain cleaning fluid	AAA-DA4-10-00-00	AAA-DA4-10-00-00	0.9983	YES
Chain - Oil	AAA-DA4-10-00-00	AAA-DA4-10-00-00	0.9974	YES
Drivetrain - Description of how it is made	AAA-DA4-00-00-00	AAA-DA4-00-00-00	0.9984	YES
Frame - Description of how it is made	AAA-DA3-00-00-00	AAA-DA3-00-00-00	0.9992	YES
Front wheel - Fault reports and isolation procedures	AAA-DA0-10-20-00	AAA-DA0-10-20-00	0.9971	YES
Gears - Description of how it is made	AAA-DA5-00-00-00	AAA-DA5-00-00-00	0.9997	YES
Handlebar - Install procedures	AAA-DA2-20-00-00	AAA-DA2-20-00-00	0.9970	YES
Handlebar - Remove procedures	AAA-DA2-20-00-00	AAA-DA2-20-00-00	0.9986	YES
Headset - Description of how it is made	AAA-DA2-30-00-00	AAA-DA2-30-00-00	0.9996	YES
Headset - Install procedures	AAA-DA2-30-00-00	AAA-DA2-30-00-00	0.9986	YES
Headset - Remove procedures	AAA-DA2-30-00-00	AAA-DA2-30-00-00	0.9981	YES
Horn - Remove and install a new item	AAA-DA3-10-00-00	AAA-DA3-10-00-00	0.9979	YES
Hubs - Clean with degreasing agent	AAA-DA5-20-00-00	AAA-DA5-20-00-00	0.9988	YES
Inner tube - Remove and install a new item	AAA-DA0-10-10-00	AAA-DA0-10-10-00	0.9979	YES
Mechs - Description of how it is made	AAA-DA5-10-00-00	AAA-DA5-10-00-00	0.9989	YES
Rear wheel - Detected fault	AAA-DA0-20-00-00	AAA-DA0-20-00-00	0.9964	YES
Rear wheel - Remove procedures	AAA-DA0-20-00-00	AAA-DA0-20-00-00	0.9986	YES
Shifters - Description of how it is made	AAA-DA5-30-00-00	AAA-DA5-30-00-00	0.9994	YES
Steering - Description of how it is made	AAA-DA2-00-00-00	AAA-DA2-00-00-00	0.9996	YES
Stem - Install procedures	AAA-DA2-10-00-00	AAA-DA2-10-00-00	0.9985	YES
Stem - Remove procedures	AAA-DA2-10-00-00	AAA-DA2-10-00-00	0.9980	YES
Tire - Check pressure	AAA-DA0-10-20-00	AAA-DA0-10-20-00	0.9980	YES
Tire - Fill with air	AAA-DA0-10-20-00	AAA-DA0-10-20-00	0.9974	YES
Tire - Remove and install a new item	AAA-DA0-10-20-00	AAA-DA0-10-20-00	0.9981	YES
Wheel - Description of how it is made	AAA-DA0-00-00-00	AAA-DA0-00-00-00	0.9986	YES
Overall	-	-	-	82.3529%

TABLE 7.3 | S1000D: 5-STAR Accuracy.

In total 34 macro structures have been semantically annotated with concepts from an ontology. The number of correctly annotated macro structures is 28. Thus, the 5-STAR accuracy as defined in the previous section is 82.3529%. The six annotations that were wrong all concerned macro structures describing the complete bike. All of these macro structures retrieved the annotation concept “AAA-000-00-00-00” instead of “AAA-D00-00-00-00”. Actually, “AAA-000-00-00-00” is a more general concept than “AAA-D00-00-00-00” and can thus be considered as more or less correct.

### 7.2.7 System Use

The S1000D Bike data set is not intended for productive usage. As part of the S1000D specification it serves as data set for evaluation purposes. The semantification process applied in this case study, however, yields a semantified data set that is runnable in a state-of-the-art semantic information system. Figure 7.8 shows an excerpt of the defined semantification pipeline in CAPLAN.

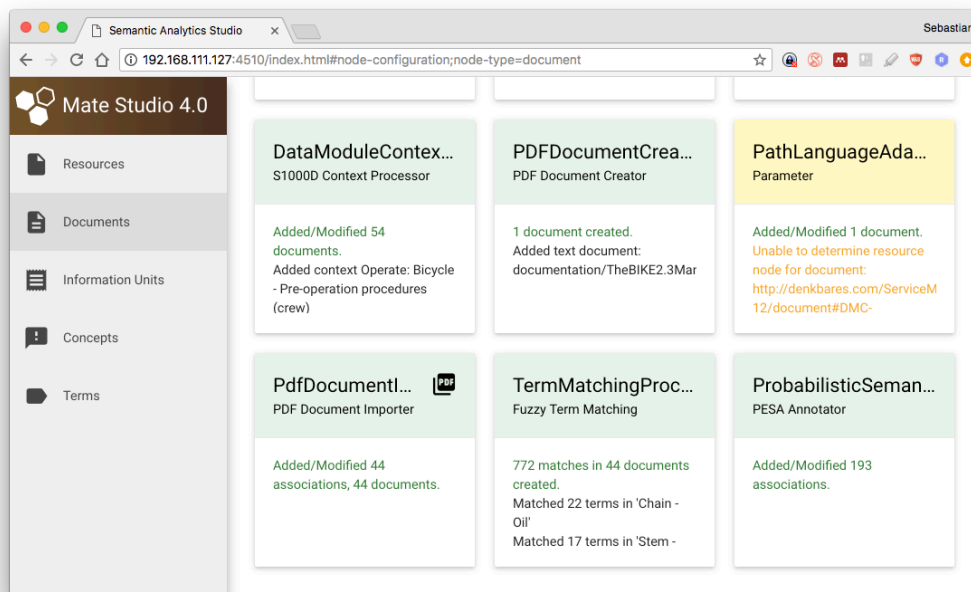


FIGURE 7.8 | S1000D: Semantification Pipeline in CAPLAN.

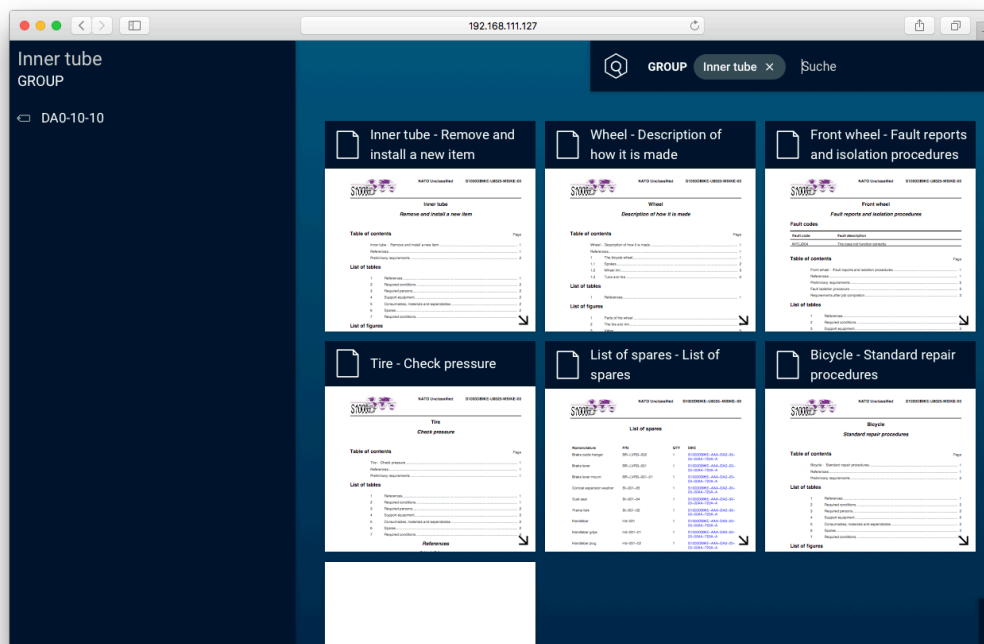


FIGURE 7.9 | S1000D: Semantification Result in the information system “Service Mate”.

Figure 7.9 shows the semantified data set in the semantic information system “Service Mate”. More precisely the screenshot shows the search results for the semantic query “Inner tube”. The top ranked search result is a module that describes the remove and install procedures for the inner tube of the S1000D bike. Additional search results are closely related to the inner tube, e.g. a descriptive module for the

wheel, fault and isolation procedures for the front wheel and check procedures for the tire.

### 7.2.8 Summary

This case study described the complete semantification of a PDF-based documentation. The semantification comprised several semantification steps. The recovery of micro and macro structures has been described and evaluated in detail. Additionally, a Subject Analysis task that was performed on the recovered macro structures has been evaluated. The micro structure recovery is based on a knowledge base that was assembled using TEKNO Studio. After minor adaptations to the automatically generated Set-Covering models the micro structure recovery yields almost perfect results (98% accuracy). Although the macro structure recovery costs are rather high from a formal perspective, the actual results are very good. The Subject Analysis tasks yields an average accuracy of 82%. This, however, is mainly based on the fact that the gold standard data only provides one single concept annotation per module and Subject Analysis method chose a more general or specialized topic in some cases. Manual inspections and the described system usage underline that the performed Subject Analysis yields very good results.

## 7.3 PI-Fan

The PI-Fan is an evaluation data set consisting of technical publications for fictive table fans. The data set is designed for benchmarking retrieval tools with respect to the PI-Mod information model.

### 7.3.1 Introduction

PI-Mod is an information model for technical documentation that is widely implemented in a variety of proprietary enterprise content management systems, especially in the German market (see Section 3.4.1). The “Pi-Fan” corpus [201] is an evaluation data set for the PI-Mod information model and comprises multilingual technical documentation for fictive table fans and has been created for benchmarking content management and content delivery systems. The corpus comprises modularized source files in Office Open XML (Microsoft Word) and DITA format. Each module is self contained and describes a certain topic of interest. A complete PDF documentation can easily be assembled from these source files. This fact makes it a perfect source for the evaluation of the presented semantification process as the data set is available in two XML-based source formats and as compiled PDF publication.

### 7.3.2 Goals and Application Scenarios

Technical documentation written according to the PI-Mod information model is already modularized. Each module contains textual content and is accompanied by metadata in forms of product and information classes. Thus, the primary application scenario of technical documents created according to the PI-Mod information model is the targeted information supply of people working with the documents, e.g. service technicians.

This case study, however, works upon the compiled PDF documentation, which is not modularized, not information typed, and not semantically annotated. This case study especially focuses on the 2-STAR semantification, i.e. the recovery of micro structures. Therefore, relevant micro structures get recovered in the PDF document. The semantified PDF-based data set is evaluated against PI-Fan source files (see Section 7.3.6).

### 7.3.3 Data Set Description

The PI-Fan data is intended for benchmarking PI-Mod compatible content management and content delivery tools. The data set comprises a technical manual for a fictive table fan. The technical manual is separated into 29 modules in Office Open XML (Word) format. The publication additionally contains 49 illustrations.

### 7.3.4 The Semantification Process

This section describes the semantification architecture used to semantify the described data set. The pipeline (see Figure 7.2) initially reads the PDF file (1-STAR rating). Subsequent processing steps aim on recovering nano (A) and micro (B) structures.

The nano structure recovery (A) is based on the open source tool pdf2xml that has been proposed by Dejean et al. [52]. The micro structure recovery (B) uses the knowledge base described in Section 7.3.5 to estimate micro structures.

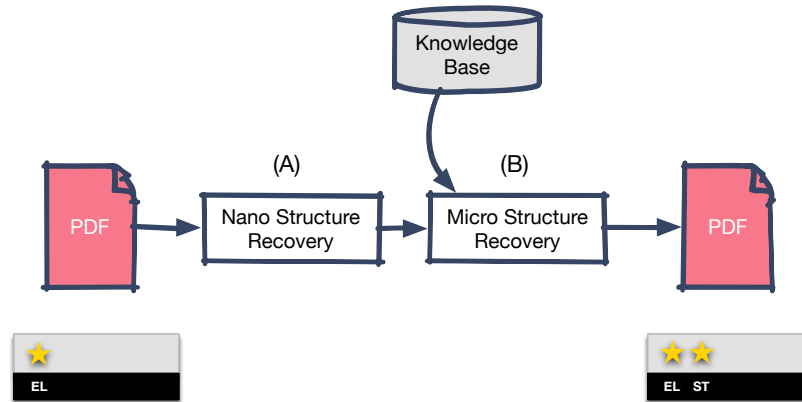


FIGURE 7.10 | PI-Fan: PDF Semantification Architecture.

### 7.3.5 Knowledge Resources

The 2-STAR semantification requires a knowledge base that contains classification knowledge for micro structures. For this case study a respective knowledge base has been assembled using TEKNO Studio. The knowledge base contains ten Set-Covering models for the following micro structure types: Caption, Caution, Heading 1, Heading 2, Heading 3, Note, Procedure, Text, Title and Warning. Each of the Set-Covering models has on average seven Set-Covering relations that reflect formatting features of the respective micro structure type.

### 7.3.6 Evaluation

This section describes the evaluation of the semantification architecture with respect to the PI-Fan data set. First the employed evaluation procedure is described. Then the employed evaluation methods are detailed. The section closes with describing respective evaluation results.

#### Evaluation Procedure

For the evaluation of the semantification results the following evaluation procedure has been applied. First, the source files in Office Open XML (Word) format have been aligned with the result of the PDF semantification pipeline. Then, the results of the 2-STAR semantification have been evaluated. The evaluation mechanisms considered the source XML files as gold standard. The evaluation methods employed are described in detail in the next section.

#### Evaluation Methods

The **2-STAR** semantification aims on the recovery of micro structures like headlines, paragraphs, or lists. The Office Open XML source files provide respective markup for these elements, i.e. they are directly accessible. The semantification architecture for the PDF file determines corresponding micro structures for the PDF document. Having micro structures available in both the source files and the PDF modules the accuracy of the micro structure recovery processor can be determined. The accuracy is defined as follows:

**Definition 7.3.1** (Micro Structure Recovery Accuracy). The micro structure recovery accuracy  $acc_{2-STAR}$  is defined as the number of correctly classified blocks



$M_{c, pdf} \subseteq M_{pdf}$  in the PDF modules with regard to all blocks  $M_{xml}$  available from source files in Office Open XML format:

$$acc_{2-STAR} = \frac{|M_{c, pdf}|}{|M_{xml}|}.$$

A block  $p \in M_{pdf}$  is correctly classified, if there is a block  $x \in M_{xml}$  with the same type and text assigned.

## Evaluation Results

In total, 79 micro structures were automatically classified using the assembled knowledge base. The classification using the automatically generated knowledge base yielded the following result:

$$acc_{2-STAR} = \frac{|M_{c, pdf}|}{|M_{xml}|} = \frac{70}{79} = 0.8861.$$

The reasons for misclassifications have been inspected manually. Formatting errors and inconsistencies in the PI-Fan documentation were identified as the main reason.

### 7.3.7 System Use

The PI-Fan data set is intended for evaluation and benchmarking purposes. Thus, the documentation is in huge parts incomplete and inconsistent. For this reason, no further semantification steps have been performed. Hence, there is no real system usage. Figure 7.11 shows the acquisition of 2-STAR classification knowledge using TEKNO Studio.

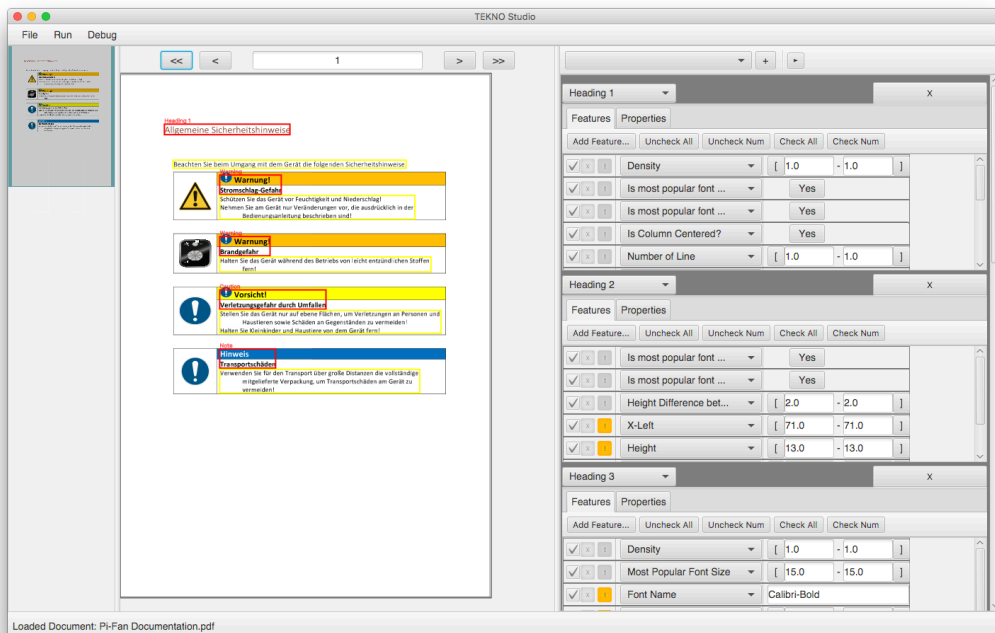


FIGURE 7.11 | PI-Fan: 2-STAR knowledge acquisition in TEKNO Studio.

### 7.3.8 Summary

This case study reported on the 2-STAR semantification of the PI-Fan data set. The PI-Fan data set is intended for benchmarking and evaluation purposes and thus incomplete and inconsistent. The 2-STAR semantification aims on recovering micro structures. In the course of this case study, classification knowledge in forms of Set-Covering Models for ten different micro structures have been defined. The classification knowledge has then been applied to the PI-Fan documentation in PDF format. The results have been evaluated against the XML source data. Although the formatting of micro structures in the PI-Fan documentation is rather inconsistent an average accuracy of 88.61% has been achieved.

## 7.4 Augmenting Spare Part Catalogues

This case study reports on a semantification architecture that is repeatedly employed. The intended usage of the semantification architecture is linking electronic spare part catalogues with technical documents.

### 7.4.1 Introduction

CATALOGCreator<sup>®2</sup> is one of the leading standard software products for electronic spare parts catalogues. Recently, the makers of CATALOGCreator<sup>®</sup> decided to further develop the software to become an information system for service technicians. In this development, linking spare part information and corresponding technical information is one important aspect. A typical usage scenario is the connection of a spare part catalogue entry with corresponding install and remove procedures. As CATALOGCreator<sup>®</sup> has a broad customer base the repeatable application of a unified semantification pipeline to similar data sets is an important success factor.

### 7.4.2 Goals and Application Scenario

This case study describes a semantification architecture that is able to link spare part information with corresponding technical documents. As existing technical documents are usually not modularized and not semantically annotated the 5-STAR semantification approach gets applied. Hence, the goal of the presented architecture is to repeatedly produce semantified data sets that are annotated/linked to spare part information. The underlying corpora of technical documents usually exist in PDF format. Therefore, the documents are subsequently semantified by running through respective semantification steps. Unlike other semantification processes the fifth step in this architecture links the modules to concepts representing entries of the spare parts catalogue. The respective ontology representing the entries of the spare parts catalogue is automatically derived from the source data of CATALOGCreator<sup>®</sup>.

### 7.4.3 Data Set Description

The semantification architecture for the linking of technical documents and spare part entries of CATALOGCreator<sup>®</sup> is intended to be repeatedly applicable to a large amount of similar data sets from different customers. Thus, a thorough data set description is not possible for this case study. Usually, a CATALOGCreator<sup>®</sup> instance comprises spare part information for a single machine. The spare part information is persisted in a relational data base and provides hierarchy information. This information can be easily transformed to an ontological representation.

### 7.4.4 The Semantification Process

The semantification process is divided into two separate pipelines (see Figure 7.12). The first pipeline extracts concepts (1) representing spare parts from CATALOGCreator<sup>®</sup>'s data base and additionally adds hierarchy information (2). The second pipeline is actually performing the semantification. Therefore, it initially reads PDF files that are usually provided with 1-STAR rating. Subsequent processing steps aim on recovering nano (A), micro (B) and macro (C) structures. Finally, a semantic annotation process (D) determines concepts from the spare parts catalogue for the

---

<sup>2</sup><https://www.tid-informatik.de>

semantic annotation of the PDF modules. The ontology used for the semantic annotation step is the result of the first pipeline that transforms data base entries to an ontological representation. The resulting PDF modules finally yield 5-STARS on the maturity schema. Additionally, these PDF modules are automatically linked to the spare parts available in CATALOGCreator<sup>®</sup>.

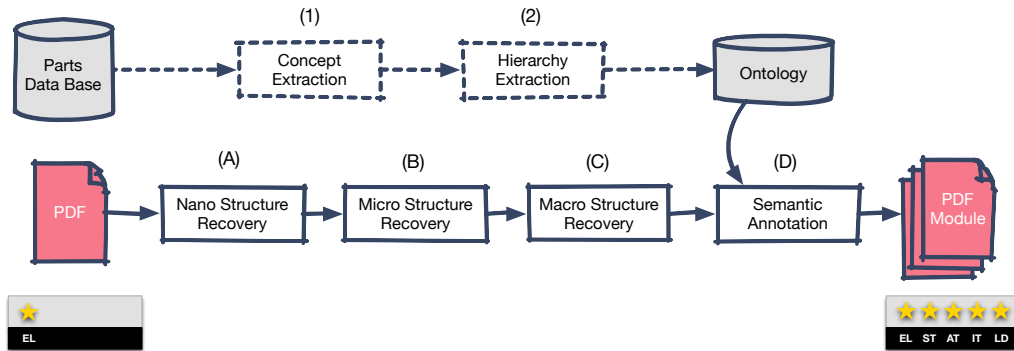


FIGURE 7.12 | Parts Catalogue: PDF Semantification Architecture.

The nano structure recovery (A) is based on the open source tool pdf2xml that has been proposed by Dejean et al. [52]. The micro structure recovery (B) is rather simple compared to other semantification architectures and concentrates on determining headlines. This can usually be achieved by exploiting the embedded table of contents of PDF documents. The macro structure recovery (C) works upon the recovered micro structures and employs the macro structure recovery mechanisms described in Section 4.5.3. The semantic annotation (D) is based on Probabilistic Explicit Semantic Analysis (PESA) described in Section 4.7.4 and uses the extracted ontology.

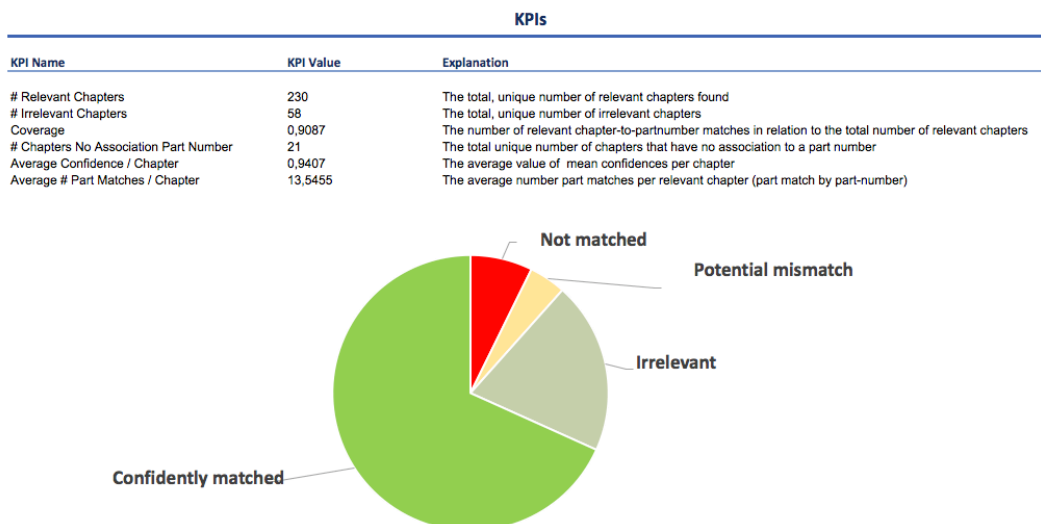


FIGURE 7.13 | Parts Catalogue: Semantification KPIs.

The probabilistic model is parametrized to incorporate existing domain knowledge. A thorough description of the parametrization is not possible for this case study as it highly depends on the actual customer data. However, the parametrization is usually an iterative process in this application scenario. Therefore, a report mechanism

has been implemented for the CAPLAN architecture that is able to provide key performance indicators (KPI) for a specific parametrization and data set. Figure 7.13 shows a sample report with the respective KPIs. In order to provide an indicator of the semantification quality the report lists how many modules of the documentation are confidentially mapped to spare parts, how many modules are considered to be irrelevant, as well as potential mismatches and missing modules.

#### **7.4.5 System Use**

The semantification process applied in this case study, yields a data set of technical documents that is directly linked to spare part instances. The semantic information system “Service Mate” has been integrated into the CATALOGCreator<sup>®</sup> application. Service Mate provides a context sensitive semantic search mechanism for spare parts. This enables CATALOGCreator<sup>®</sup> to provide the user technical documents that are relevant for a specific spare part. Figure 7.14 shows a screenshot of CATALOGCreator<sup>®</sup>, the integrated Service Mate module and search results for a spare part called “Hohlwelle”. A search for relevant technical documents is executed automatically when the user selects a spare part.

#### **7.4.6 Summary**

This case study reported on a semantification pipeline that is intended for repeated integration tasks. The corresponding application scenario described CATALOGCreator<sup>®</sup>, a standard application for electronic spare part catalogues, that has been extended with a semantic search engine. The goal of the semantic search engine is to provide relevant technical documents in the context of a specific spare part. Therefore, a semantification architecture has been implemented that allows to link technical documents to spare part instances. The semantification comprised several semantification steps that aim on recovering nano, micro and macro structures. Then, a Subject Analysis task is performed on the recovered macro structures in order to actually link spare parts and modules. A formal evaluation has not been performed on this data set. However, multiple corpora of technical documents from different customers have already been connected to the respective electronic spare part catalogues.

**CATALOGcreator**

HOME | KATALOG | SCHALTPLAN | WARENKORB | SUCHE | DOKUMENTE | FAVORITEN | SERVICEMATE

Sprache des Kataloginhalts: Deutsch

Navigation:
 

- Motor drehgestell (12128618)
- Rahmen MDG (12000429)
- Prüfmaerstufe kpl. MDG GMT, KONI (12136447)
- Sekundaerstufe MDG Filtr. 25/70, 2/4 SD, GMT, KONI (12139827)
- Triebradsatz mont. NSR (12143248)
- Triebradsatz MDG NSR (12143257)
- Achsantrieb kpl (12121953)
- Motor kpl. TSA012203R1 (12143731)
- Getriebe - V160 mit Keilpaketkupplung - Kurz (12143416)
- Keilpaketkupplung Kurze Variante, KPK D570-2-2/6 (12082292)
- Getriebe - V160 ohne Keilpaketkupplung (12140028)**
- Getriebegehäuse Mit Rückfahrführchen (12114619)
- Getriebedeckel Grundierung RAL 1015 (12003994)
- Ruecklaufrohr kpl. (12078689)

Getriebe - V160 ohne Keilpaketkupplung (12140028)

3D PDF

Das Media-Element wird von Ihrem Browser nicht unterstützt.  
Weitere Informationen finden Sie in den Systemvoraussetzungen.

**Bauteil-Details**

Informationen  
 Artikelnr.: 12003490  
 Bezeichnung: Hohlwelle

Stückliste  
 12003490\_Vers.009...Stueckliste.deu  
 19.92 KB

Notizen  
 Keine Notiz in dieser Sprache vorhanden

ServiceMate Dokumente

Pos.	Artikelnr.	Bezeichnung	Ersatzteil	Menge auf Stückliste	Mengeneinheit
0108	12143622	Gewindestift I-8x1 und Keilglocke BN8010Z M10x1,0 precole	x	24	Stk
0105	12137327	Halbrundkerbriegel BN688 3x8		4	Stk
0402	12003490	Hohlwelle		1	Stk
0403	12003342	Keinellotlager E-801574.TR1 d=304.8 D=393.7 B=50.8		2	Stk
0405	12112647	Labrynthdeckel Hohlwelle		2	Stk
0216	12012559	Labrynthdeckel Welle 1		1	Stk

ServiceMate Dokumente

- Hohlwelle
- 5.1.2.7 Axialspiel der Hohlwelle messen
- 5.1.2.9 Axialspiel der Hohlwelle erneut messen
- 5.1.3.1 Zylinderrollenlager montieren
- 8.1 Spezielle Hilfsmittel
- 3.1 Technische Daten Radsatzgetriebe
- 4.1 Radsatzgetriebe transportieren
- 5.1.2.3 Motorsseitiges Keigelrollenlager montieren

powered by CATALOGcreator®

FIGURE 7.14 | Parts Catalogue: Semantification Result.

## 7.5 Earth Moving Technology

This case study reports on an actual semantification project for earth moving technology. The project goal required two tasks to be achieved: (1) split a corpus of legacy technical documents in PDF format to information units, and (2) extract metadata from these documents.

### 7.5.1 Introduction

This case study reports on an actual semantification project for a German mechanical engineering company for earth moving technology. The semantification affected a large corpus of technical documentation (more than 40 GB of PDF documents) covering different vehicles from the field of earth moving technology. The documentation not only covers different skill levels of users, configurations and aspects of products, but also targets different markets and their respective languages, e.g., German, English, French, and Russian. Hence, documentation exists for each variant of a machine and different markets. In order to provide efficient customer support, fast and effective access to relevant information becomes a critical success factor. Therefore, the respective mechanical engineering company introduces a semantic information system.

Amongst other data pools the semantic information system considers the large corpus of technical documents. However, large fractions of the corpus comprises legacy documentation, i.e., technical documents that are not prepared for the usage in semantic systems. A manual modularization and annotation of the legacy documents requires an in-depth analysis of the original content by humans, which is usually error-prone, time-consuming, and very cost-intensive for large-scale corpora. Therefore, a semantification architecture has been applied that successfully transforms 1-STAR legacy documents to semantically prepared 5-STAR modules.

### 7.5.2 Goals and Application Scenario

The underlying semantic information system offers service technicians targeted access to required information. Therefore, the semantic information system provides different facets that allow the user to successively detail a search query. However, in order to provide access to relevant information, the information system requires technical documents to be semantically prepared. Hence, the goal of the semantification project was the transformation of a large-scale PDF-based data set into a representation that is compatible with a semantic information system. The PDF resources, therefore, have been semantically prepared by first splitting them to reasonable modules which then have been annotated with metadata that has been extracted from discovered Core Documentation Entities.

### 7.5.3 Data Set Description

The technical document corpus in focus contains several thousand documents and covers different vehicles. The technical documents are either service manuals or maintenance manuals. The documents address different target groups and are provided in PDF format. Source files are either not available or exist in a proprietary format. The metadata that is necessary for the semantic annotation and indexing of information units is encapsulated as written text in the respective documents.

### 7.5.4 The Semantification Process

This section describes the semantification process used to semantify the described data set. The section first introduces the semantification architecture and then describes the involved knowledge resources.

#### Semantification Architecture

The pipeline (see Figure 7.15) initially reads the PDF file (1-STAR rating). Subsequent processing steps aim on recovering nano (A), micro (B) and macro (D) structures. A Core Documentation Entity discovery is performed on the recovered micro structures. Discovered Core Documentation Entities are exploited in order to extract metadata. The extracted metadata is used as input for a semantic annotation processor (E) that adds the respective information to the PDF modules. These PDF modules finally yield 5-STARS in the maturity schema and are semantically prepared for the usage in a semantic information system.

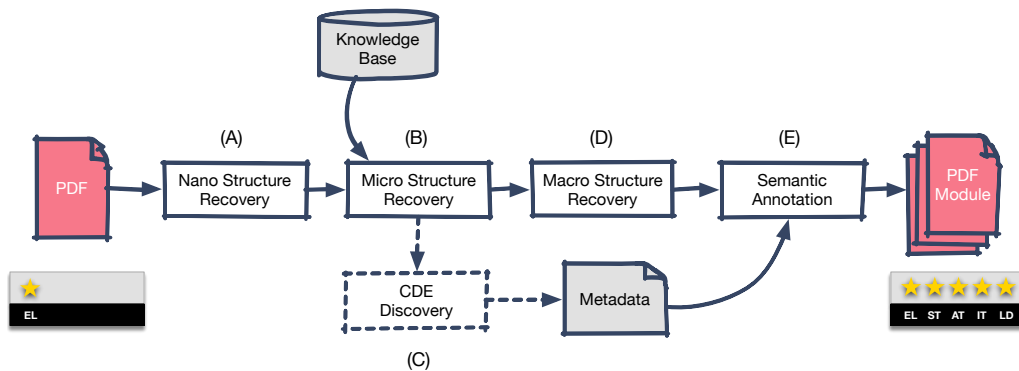


FIGURE 7.15 | Earth Moving Technology: Semantification Architecture.

The nano structure recovery (A) was performed using the open source tool pdf2xml [52]. A knowledge-based micro structure recovery was employed that identified different kinds of micro structures (B) and prepared the Core Documentation Entity discovery (C). The macro structure recovery (D) works upon the recovered micro structures and employs the macro structure recovery mechanisms described in Section 4.5.3. The semantic annotation (E) simply adds the extracted metadata to respective PDF modules.

The customer required for each module the following meta information:

- **Type:**  
The type of the vehicle; usually an alphanumeric code.
- **Model:**  
Further specification of the vehicle; usually a variant of the vehicle type.
- **Drawing Number:**  
Drawing Numbers are used to differentiate different versions of a model.
- **Serial Number:**  
The unique serial number of a single vehicle.

The meta information was spread over the respective documents. Some of the information, like type and drawing number, were defined on the title page, while others,



like model and serial number, were defined on special index pages. Other metadata, like the title of the chapter, needed to be extracted from the respective chapters. Figure 7.16 illustrates the metadata extraction from the different document locations.

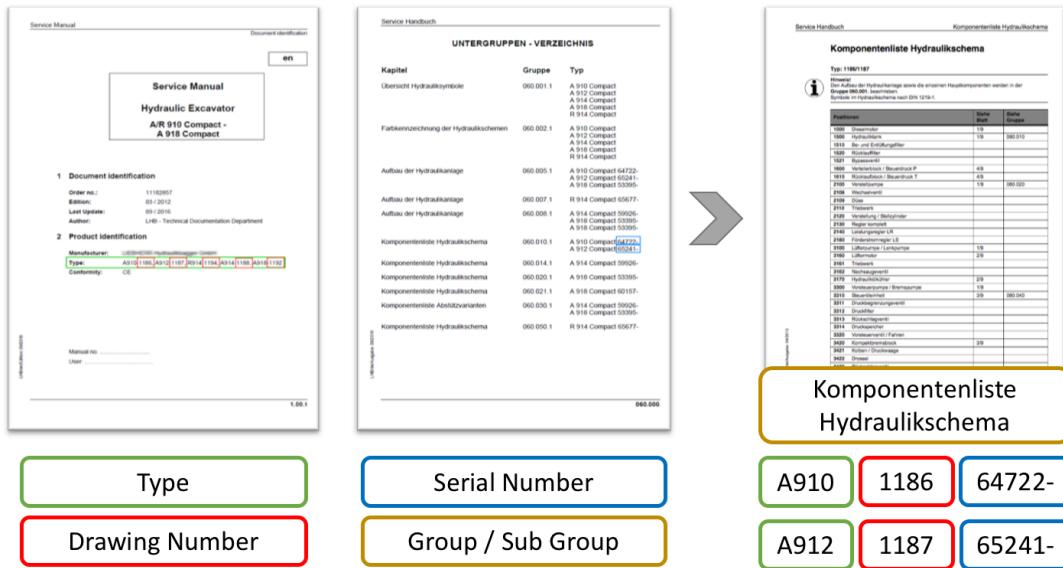


FIGURE 7.16 | Earth Moving Technology: Extraction.

In order to provide the customer insights into the semantification quality CA-PLAN’s report mechanism was employed. The respective report lists all recovered modules and the identified metadata. Figure 7.17 shows an excerpt of such a report.

**Results**

File	Title	Type	Drawing	Model	Serial
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_10.10.pdf	10.10 Kugeldrehkranz	A 309	1035	LI-TCD	40998-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_10.10.pdf	10.10 Kugeldrehkranz	A 311	1036	LI-TCD	41128-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_10.10.pdf	10.10 Kugeldrehkranz	A 312	1037	LI-TCD	36220-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_10.10.pdf	10.10 Kugeldrehkranz	A 314	1039	LI-TCD	30694-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_10.10.pdf	10.10 Kugeldrehkranz	A 316	1041	LI-IND-TCD	29503-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_10.10.pdf	10.10 Kugeldrehkranz	A 316	1041	LI-TCD	29120-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_10.10.pdf	10.10 Kugeldrehkranz	R 313	1038	LI-TCD	39171-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_10.10.pdf	10.10 Kugeldrehkranz	R 317	1043	LI-IND-TCD	41528-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_10.10.pdf	10.10 Kugeldrehkranz	R 317	1043	LI-TCD	30019-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_11.20.pdf	11.20 Achsverteilergetriebe 2 HL 270 / 290	A 309	1035	LI-TCD	40998-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_11.20.pdf	11.20 Achsverteilergetriebe 2 HL 270 / 290	A 311	1036	LI-TCD	41128-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_11.20.pdf	11.20 Achsverteilergetriebe 2 HL 270 / 290	A 312	1037	LI-TCD	36220-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_11.20.pdf	11.20 Achsverteilergetriebe 2 HL 270 / 290	A 314	1039	LI-TCD	30694-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_11.20.pdf	11.20 Achsverteilergetriebe 2 HL 270 / 290	A 316	1041	LI-IND-TCD	29503-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_11.20.pdf	11.20 Achsverteilergetriebe 2 HL 270 / 290	A 316	1041	LI-TCD	29120-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_11.32.pdf	11.32 HBGV-Block für 2 HL 270 / 290	A 309	1035	LI-TCD	40998-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_11.32.pdf	11.32 HBGV-Block für 2 HL 270 / 290	A 311	1036	LI-TCD	41128-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_11.32.pdf	11.32 HBGV-Block für 2 HL 270 / 290	A 312	1037	LI-TCD	36220-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_11.32.pdf	11.32 HBGV-Block für 2 HL 270 / 290	A 314	1039	LI-TCD	30694-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_11.32.pdf	11.32 HBGV-Block für 2 HL 270 / 290	A 316	1041	LI-IND-TCD	29503-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_11.32.pdf	11.32 HBGV-Block für 2 HL 270 / 290	A 316	1041	LI-TCD	29120-
output/SHB_A309-R317-LI-TCD_de_A309-R317-LI-TCD_de_11.60.pdf	11.60 Fahrwerksgetriebe FAT 350	R 313	1038	LI-TCD	39171-

FIGURE 7.17 | Earth Moving Technology: Semantification Report.

### 7.5.5 Knowledge Resources

The micro structure recovery and Core Documentation Entity discovery are the key elements of the aforementioned semantification architecture. Therefore, the corpus was manually pre-classified into disjoint partitions. For each partition a d3web knowledge base was defined. The knowledge acquisition necessary for the definition of respective knowledge bases was performed using TEKNO Studio. TEKNO Studio uses

the blockification mechanism of lapdf<sup>3</sup>. As the blockification results varied for the different partitions of the corpus the blockification engine has been significantly extended. The respective extension introduced a new blockification mechanism and allows for the corpus-based configuration of the block detection. In summary, eleven different block detection configurations have been defined for this project. Each of these configurations specify the average distance (left, right, top, bottom) between words. The respective knowledge bases contain Set-Covering models for the following classes:

- Model (Machine)
- Type
- Drawing Number
- Serial Number
- Chapter Number
- Chapter Title
- (Document) Version

### 7.5.6 System Use

The semantified technical documentation is used in a semantic information system for after sales service. The metadata attached by the described semantification architecture enable a semantic search that works upon different taxonomies that describe products and components.

Figure 7.18 and Figure 7.19 show the acquisition of 2-STAR classification knowledge using TEKNO Studio. The screenshots show classified document structures that are spread over different title and index pages. Respective Set-Covering Models are defined on the right panel.

---

<sup>3</sup><https://github.com/BMKEG/lapdf<sup>3</sup>>

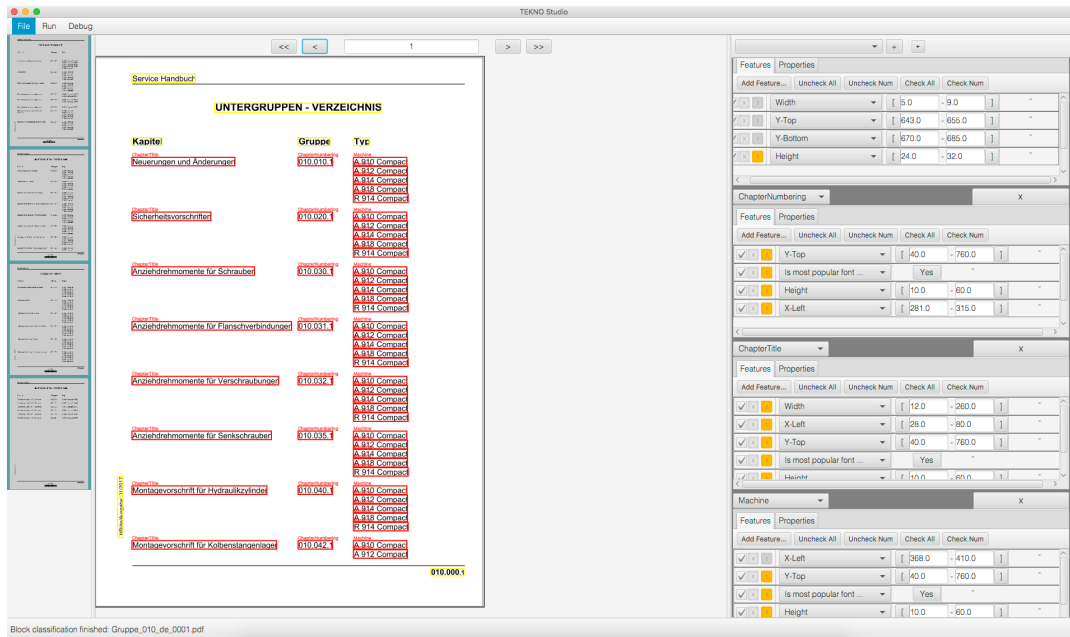


FIGURE 7.18 | Earth Moving Technology: 2-STAR knowledge acquisition in TEKNO Studio (1/2).

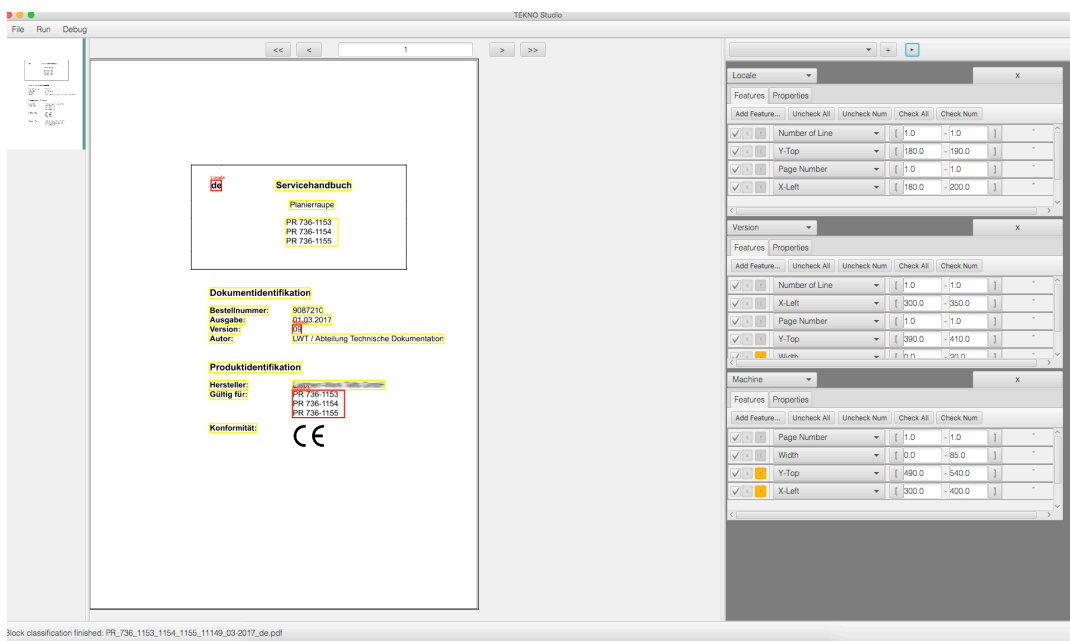


FIGURE 7.19 | Earth Moving Technology: 2-STAR knowledge acquisition in TEKNO Studio (2/2).

### 7.5.7 Summary

This case study reported on an actual semantification project in the domain of earth moving technology. In contrast to other case studies, the presented semantification architectures focused on the extraction of metadata from Core Documentation Entities. The extraction results were subsequently used as semantic annotations that enable semantic search over the respective corpus. The underlying Core Documentation Entities were recovered using the 2-STAR block classification techniques described

in Section 4.4.3. The presented semantification architecture allows to process large corpora of technical documents with reliable results. Reports are used to evaluate the semantification quality and underline the practical applicability of the presented 5-STAR semantification approach. The technical documents that have been semantified using the presented semantification architecture are available in a modern semantic information system.

## 7.6 Special Purpose Vehicles: Ontology Population

This case study reports on an ontology population project in the domain of special purpose vehicles. The case study presents a novel method to populate technical ontologies from document structures.

### 7.6.1 Introduction

Semantic information systems promise a number of advantages, like problem-oriented access to information, automated combination and aggregation of information resources, and the machine- and equipment-aware selection of appropriate information resources. These benefits are realized by employing an ontology that describes the technical breakdown of the machinery and their inter-relations. However, such ontologies do not always exist and the manual construction is time-consuming and cost-intensive. This case study presents an automated ontology population method that was successfully applied multiple times in the domain of special purpose vehicles.

### 7.6.2 Goals and Application Scenario

The goal of the presented case study is the population of multiple technical ontologies that represent special purpose vehicles. The ontologies are intended for a subsequent usage in a state-of-the-art semantic information system. Therefore, the elements of the ontologies are connected with information resources of the technical documentation. This case study, however, describes the exploitation of Core Documentation Entities in order to populate the initial ontologies. The ontology structure (classes and properties) was predefined and is the same for all vehicles. Instance data representing the respective vehicles and its components, however, were missing.

### 7.6.3 Data Set Description

The technical document corpus in focus contains about 6,000 pages of technical documentation for each vehicle. The contents are spread over 10 documents per vehicle. Each document has up to 1,000 pages and is of a certain type, e.g. repair manual, operation manual, spare parts or trouble shooting procedures. The documents address different target groups ranging from maintenance staff to end users what influences the structure and the level of detail. The documents are provided in PDF format. Source files are either not available or exist in proprietary and thus inaccessible formats.

### 7.6.4 Ontology Population Architecture

For the population of the respective vehicle ontologies an ontology population architecture has been defined. The architecture is similar to the semantification architectures that had been presented in other case studies. In contrast to the semantification architectures the ontology population architecture aims on exporting concept instances and relations between them. The pipeline depicted in Figure 7.20 initially reads the PDF file (1-STAR rating). Subsequent processing steps aim on recovering nano (A) and micro (B) structures. Then, for the whole document an information type gets determined (C). This is the basis for discovering Core Documentation Entities (D). A subsequent process assembles a complete concept hierarchy from the discovered Core Documentation Entities.

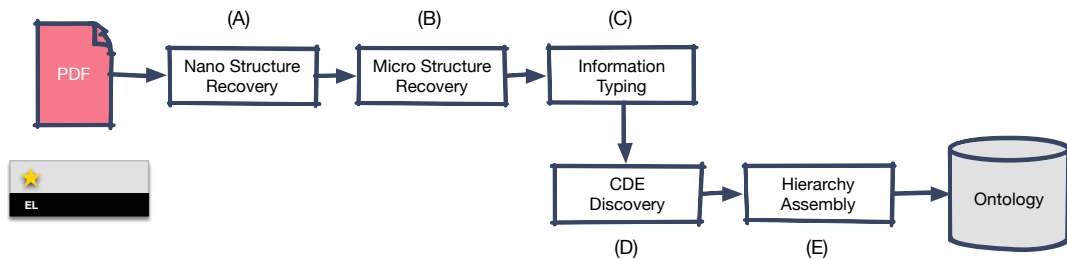


FIGURE 7.20 | Special Purpose Vehicles: Ontology Population Architecture.

The nano structure recovery (A) was performed using the open source tool pdf2xml that has been proposed by Dejean et al. [52]. A basic micro structure recovery (B) was employed that identified captions of images and corresponding legends. The micro structure recovery (B) was based on manually assembled annotation rules that were defined using the Open Source Tool Apache UIMA Ruta<sup>4</sup>. An information type was defined manually for each document. The Core Documentation Entity discovery was based on an Apache UIMA pipeline that consumed the results of the applied annotation rules.

### Core Documentation Entities

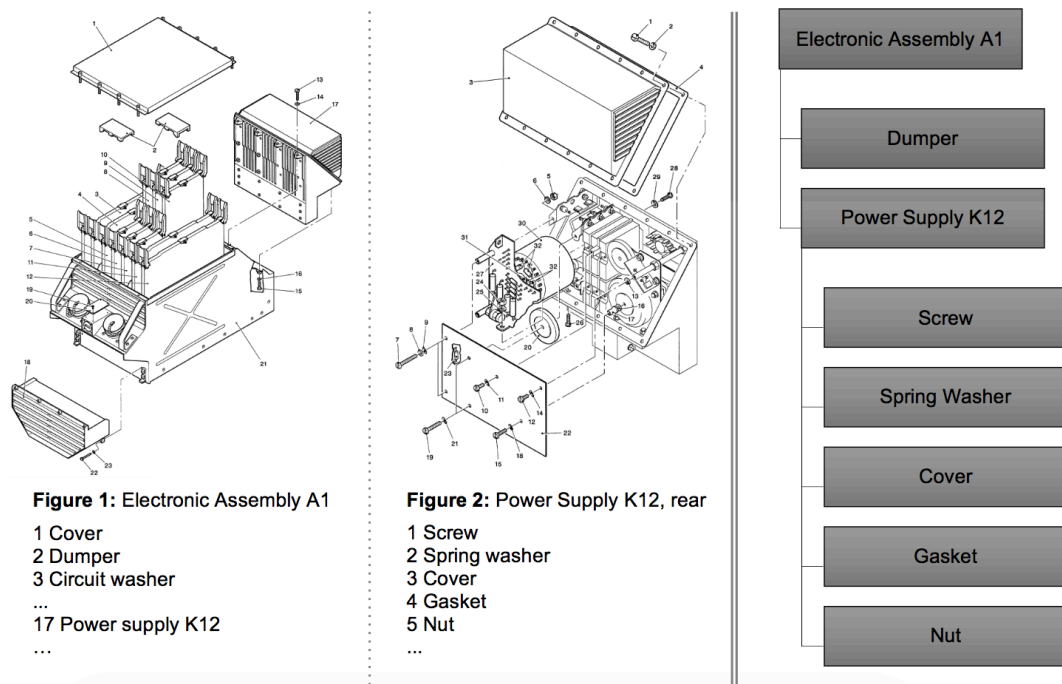


FIGURE 7.21 | Special Purpose Vehicles: Document Components and Concepts.

Interweaving structural and rhetorical representations gives access to the most important aspects of technical documents – Core Documentation Entities (see Section 3.3.3). For this case study, component overviews were the most important Core

<sup>4</sup><http://uima.apache.org/ruta.html>

Documentation type. Component overviews can typically be found in sections describing the machine and usually consist of an exploded-view drawing and an associated list of labels, product numbers, etc.:

$$\{b \in M \mid \text{type}('OrderedList', b) \wedge \text{type}('Description', b) \wedge \exists a \in M : (\text{next}(a, b) \wedge \text{type}('Figure', a)) \}$$
 (7.1)

Figure 7.21 shows an example of a component overviews and an excerpt of the resulting technical ontology.

### Constraint-based Construction of Technical Knowledge Organizations

For the construction of the respective vehicle ontologies from Core Documentation Entities (component overviews) the following process was applied:

- Partial hierarchy extraction from document components
- Entity unification
- Constraint problem generation and constraint resolution

Each of these steps will be described in more detail in the following.

#### Extracting Partial Hierarchies from Document Structures

Given a corpus of 4-STAR technical documents the direct access to *Core Documentation Entities* like repair instructions or component overviews is possible. In order to build up extensive technical ontologies partial hierarchies need to be extracted from all relevant document components. The extraction task itself can be realized as a simple query over the semantically represented corpus of technical documents.

Referring to the example depicted in Figure 7.21 partial hierarchies would be extracted for the components "Electronic Assembly A1" and "Power Supply K12". The extracted partial hierarchies (Figure 7.22) would comprise concept candidates for these two components and "part of" relations to their respective sub-components (also represented as concept candidates). The partial hierarchy describing the component "Electronic Assembly A1" here would also include a sub-component denoted "Power Supply K12" that is further detailed in another partial hierarchy. However, this further detailing is not yet semantically represented.

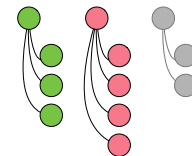


FIGURE 7.22 | Partial hierarchies.

#### Unification of Concept Candidates

In technical documents it is common practice that descriptions of components are iteratively detailed or spread over several paragraphs, e.g., because of different views revealing varying subsets of sub-components and their function. This typically results in duplicate occurrences of concept candidates in the extracted partial hierarchies. These duplicates need to be semantically unified (see Figure 7.23). The unification of duplicate concept candidate occurrences is a classical ontology matching [97] / ontology mapping [41] / ontology alignment [57]

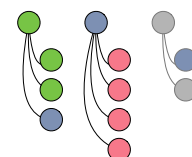


FIGURE 7.23 | Unified concepts.

task. In this case study, the unification of concept candidates has been realized using heuristic scoring rules [159] operating upon a set of concept similarity features like title of the concepts, common children and the occurrence of the corresponding Core Documentation Entity in the document.

With regard to the example depicted in Figure 7.21 the occurrences of "Power Supply K12" would be unified in this process step. Figure 7.23 shows three unified occurrences, i.e. the occurrence of "Power Supply K12" as a sub-component of "Electronic Assembly A1" (hierarchy on the left), the occurrence describing the rear view (hierarchy in the middle) and another arbitrary occurrence (hierarchy on the right).

### Constraint-based Hierarchy Construction

Now, given partial hierarchies with unified concept candidates the task is to construct a complete hierarchy. However, the scenario depicted in Figure 7.23 shows that simply connecting unconnected nodes (e.g. the root of the reddish tree) to a unified occurrence in another partial hierarchy is not possible. In the depicted scenario two possible unified occurrences exist for the respective node. For this case study a constraint-based approach for the construction of a complete technical knowledge organization has been employed. Therefore, the partial hierarchies including the already computed unifications are used to formulate a constraint-based planning problem.

A set of unified concept candidates get formally defined as  $C = \{c_1, c_2, \dots, c_n\}$ . An identity relation is defined as a tuple  $i = (parent, child)$  with  $parent \in C$  and  $child \in C$  and serves as planning entity. The  $child$  element in the relation serves as planning variable, i.e., the element that will be switched while solving the planning problem. The set of all identity relations  $I = \{i_1, i_2, \dots, i_n\}$  defines the potential search space of the planning problem. A solution of the planning problem is a child assignment for each identity relation  $i \in I$ .

In order to construct a complete and consistent technical ontology a set of constraints is defined, which a solution has to fulfill. The constraints define the conditions that the child assignments of all identity relations  $i \in I$  must not violate. Typical examples of such constraints comprise the avoidance of cycles or duplicate assignments. Standard optimization algorithms can be utilized to solve the formulated constraint satisfaction problem. For this case study, the constraint satisfaction problem has been formulated and solved using the OptaPlanner<sup>5</sup> framework. From all computed assignments the solution which violates the least constraints is chosen. Computed solutions have been exported to the open source ontology engineering tool KnowWE<sup>6</sup> [10]. The remaining violations can there be reviewed and corrected by a human expert / ontology engineer.

Referring to the example depicted in Figure 7.21 at the end of this process step the partial tree further detailing the component "Power Supply K12" would be connected to the occurrence "Power Supply K12" in the partial hierarchy of the "Electronic Assembly A1". The resulting identity relation would be represented as  $i = ("Electronic Assembly A1/Power Supply K12", "Power Supply K12")$ .

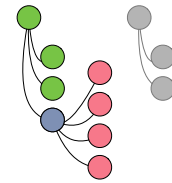


FIGURE 7.24 | Concept hierarchy.

<sup>5</sup><https://www.optaplanner.org>

<sup>6</sup><https://www.d3web.de>



### 7.6.5 System Use

The ontology population process described in this case study yields technical ontologies for the respective vehicles. Figure 7.25 shows an excerpt of exported hierarchy information in KnowWE [12]. The depicted markup is automatically compiled into an ontology that completely describes the extracted technical knowledge organization. The resulting vehicle ontologies comprise more than 1,000,000 triples.

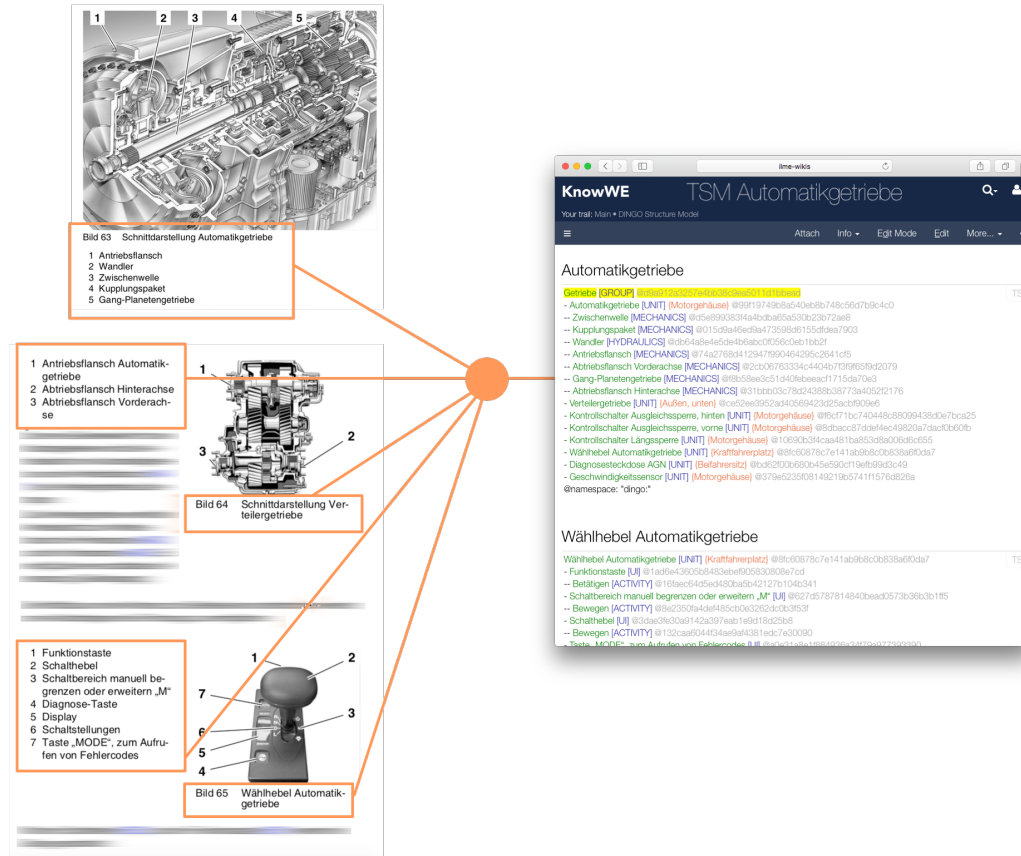


FIGURE 7.25 | Special Purpose Vehicles: Exported ontology information in KnowWE.

The ontologies are also the basis for building corresponding semantic information systems. Figure 7.26 shows a semantified data set for one of the vehicles in the semantic information system “Service Mate”. More precisely the screenshot shows the search results for the semantic query “Getriebe” (German term for “transmission”). The search results show different chapters of the corresponding technical documentation that are considered to be relevant for the respective query. The semantic summary on the left lists hierarchy information (subcomponents) which are stored in the underlying ontology.

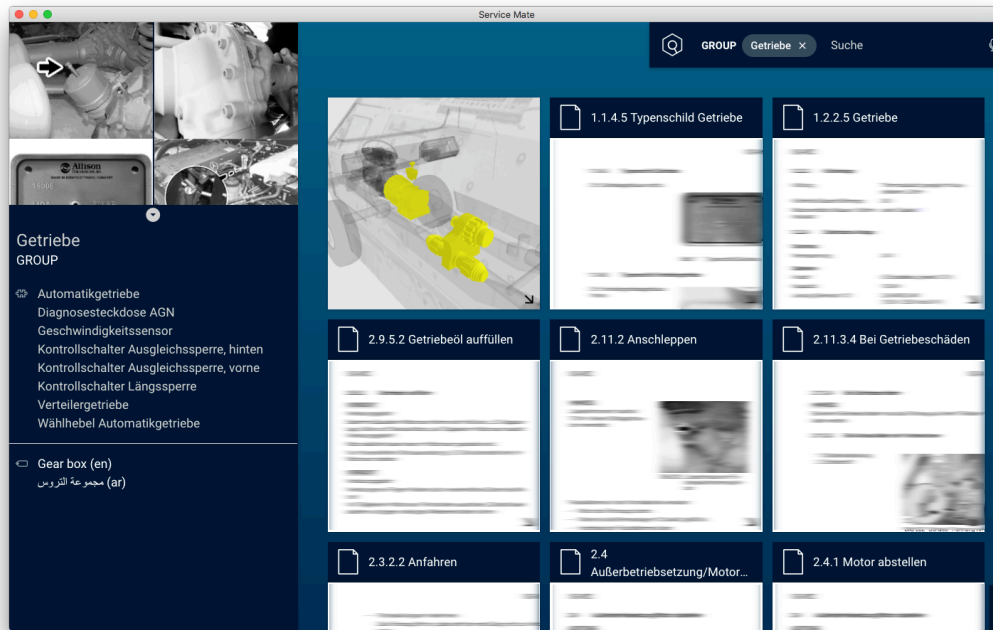


FIGURE 7.26 | Special Purpose Vehicles: Resulting data set in Service Mate.

### 7.6.6 Summary

This case study reported on an ontology population project for multiple special purpose vehicles. The ontology population was realized on the basis of semantified document structures — Core Documentation Entities. In order to be able to access these Core Documentation Entities a process comprising several semantification steps has been applied. In contrast to the other case studies the semantification process did not aim on semantifying documents but extracting information that might be useful for populating technical ontologies. Therefore, the contents of Core Documentation Entities of type `ComponentOverview` have been exploited to extract partial component hierarchies. These partial hierarchies were then assembled to a complete structure. Therefore, the hierarchy assembly task has been transformed to a planning problem. Standard tools for constraint-solving problem were used to find a solution which violates the least constraints. Then, this result has been exported to the semantic wiki KnowWE for review purposes. The resulting ontologies were the basis for the assembly of several semantic information systems.

## 7.7 Harvesting Technology

This case study reports on an actual semantification project for harvesting machines. The task was to split a corpus of legacy technical documents to information units and the subsequent determination of the most important concepts (topics) from an already existing ontology.

### 7.7.1 Introduction

This case study reports on an actual semantification project for a German mechanical engineering company for harvesting technology. The semantification affected a large corpus of technical documentation (up to 12,000 pages for a single machine). The documentation not only covers different skill levels of users, configurations and aspects of a product but also targets different markets and their respective languages. Hence, documentation exists for each variant of a machine and every relevant market. In order to provide efficient customer support fast and effective access to the relevant information becomes a critical success factor.

Amongst other data pools the semantic information system considers the large corpus of technical documents. The machinery of this company has a product-life-cycle of several decades. Authoring guidelines ensure that newly created documents are semantically prepared, i.e. they are authored in forms of modules and get annotated with respective metadata. However, large fractions of the corpora comprise legacy documentation, i.e. technical documents that are not prepared for the usage in a state-of-the-art semantic information system. A manual modularization and annotation of the the legacy documents requires an in-depth analysis of the original content by humans, which is usually error-prone, time-consuming, and very cost-intensive for such a large scale corpus. Therefore, a semantification architecture has been applied that successfully transforms 1-STAR legacy documents into semantically prepared 5-STAR modules.

### 7.7.2 Goals and Application Scenario

The underlying semantic information system offers service technicians targeted access to the required information. Therefore, the semantic information system supports users in formulating a search query and subsequently searches for relevant information. However, in order to provide access to relevant information, the information system requires technical documents to be semantically prepared. Hence, the goal of the semantification project was the transformation of a large-scale PDF-based data set to a representation that is compatible with a semantic information system (see Section 7.7.7). The PDF resources, therefore, have been semantically prepared by first splitting them into reasonable modules which then have been annotated with concepts from an externally provided ontology.

### 7.7.3 Data Set Description

The technical document corpus in focus contains about 9,000 documents, covering different machines. Each document has up to 2,000 pages and is of a certain type, e.g., repair manual, operation manual, circuit diagram or installation guide. The documents address different target groups ranging from maintenance staff to end users, this in turn influences the structure and the level of detail. The documents are provided in PDF format. Source files are either not available or exist in a proprietary format.

### 7.7.4 The Semantification Process

This section describes the semantification process used to semantify the described data set. First, the semantification architecture is explained. Then a detailed description of the involved knowledge resources follows.

#### Semantification Architecture

The pipeline depicted in Figure 7.27 initially reads the PDF file (1-STAR rating). Subsequent processing steps aim on recovering nano (A), micro (B), and macro (C) structures. These recovered structures already allow to split the PDF into smaller modules. Finally, a semantic annotation processor (D) determines concepts from an external ontology for the semantic annotation of the PDF modules. These PDF modules finally yield 5-STARS on the maturity schema and are semantically prepared for the usage in a semantic information system.

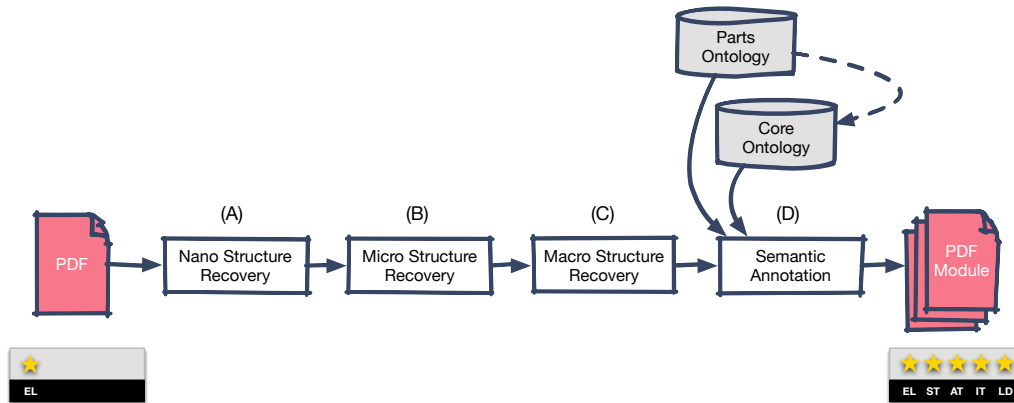


FIGURE 7.27 | Harvesting Technology: Semantification Architecture.

Nano structure recovery (A) was performed using the open source tool pdf2xml [52]. A basic micro structure recovery (B) was employed that identified headlines on different levels as well as header and footers. The micro structure recovery (B) was based on heuristics over a statistical distribution of formatting information. The macro structure recovery (C) works upon the recovered micro structures and employs the macro structure recovery mechanisms described in Section 4.5.3. The semantic annotation (D) was based on Explicit Semantic Analysis (ESA) described in Section 4.7.3 and uses a manually assembled ontology.

SPARQL [83] queries have been employed to extract concepts (assemblies) and terms. Then a dictionary-based entity recognition algorithm annotated all occurrences of terms extracted from the core and parts ontologies. The Explicit Semantic Analysis used a semantic interpreter with domain-specific weights to identify the main subjects of each module. The  $\langle k_x \rangle$  values (see Section 4.7.3) indicating the strength of the association between term  $t_z$  and concept  $s_x$  were computed differently for assembly and part terms. For terms extracted from the core ontology the weight was defined as:

$$k_x = \frac{1}{\#edges \text{ between concepts}},$$

i.e., the label of the concept in focus will get the maximum weight of 1, which means that this label indicates the concept best. Predecessors and successors in the

assembly hierarchy got lower weights, e.g., the parents and children got the weight 0.5, grandparents and grandchildren the weight 0.33.

This approach was not feasible for terms from the parts ontology, because there are parts that are semantically different but have the same label (e.g. “valve” or “screw”). The more parts have the same label, the less suitable they are for the inference of a particular concept, i.e., their weight should be adapted accordingly. Thus, the weight for terms from the parts ontology were defined as:

$$k_x = \frac{1}{\text{concept frequency}},$$

where *concept frequency* is the number of concepts that have a part represented by a particular label. This procedure shifts the focus from concepts to labels for terms from the parts ontology. The maximum weight of 1 is assigned to parts that have a unique label and are built in only one assembly. Parts with common labels that are used in a variety of assemblies get lower weights, e.g. parts with the label “screw” are built in more than 500 assemblies, so the weight is as low as 0.002.

### 7.7.5 Knowledge Resources

A domain ontology described in OWL [89] as well as a decent terminology has already been available in the company and were supplied in forms of two ontologies serialized in the Turtle [15] format. The first ontology describes relations of assemblies, products, and machines, e.g., that the cylinder block assembly is part of the engine assembly, which itself is a part of a certain product or machine – in the following this ontology is referred to as *core ontology*. The second ontology describes in detail which parts are used in a special assembly, e.g., that a certain valve is part of the cylinder head – the ontology is called *parts ontology*. Assemblies and parts have had labels attached as literals using the RDFS property `rdfs:label` and language attributes.

### 7.7.6 Evaluation

This section describes the evaluation of the employed semantification architecture with respect to the corpus of technical documents for harvesting machines. The description comprises the employed evaluation architecture and evaluation methods as well as respective evaluation results.

#### Evaluation Architecture

The evaluation of the presented semantification project covers the performance regarding the semantic annotation (5-STAR) of the information units, i.e. the identification of the main subject. No explicit evaluation of the modularization (3-STAR) has been performed. However, samples always confirmed a correct segmentation and the customer did not report any problems. As described above no training or test data was supplied by the customer, so documents that were reviewed by domain experts using the proposed review tool (cf. Section 4.7.5) were used to perform the evaluation. This allowed to measure different key performance indicators, ranging from precision, recall, and f-measure to the number of corrections that needed to be made by domain experts. Additionally, the time needed for the correction of the automatically generated results has been measured for a couple of chapters using the proposed review tool.

For the evaluation an evaluation architecture (see Figure 7.28) was set up. First, a random sample was generated from the set of processed documents. These documents

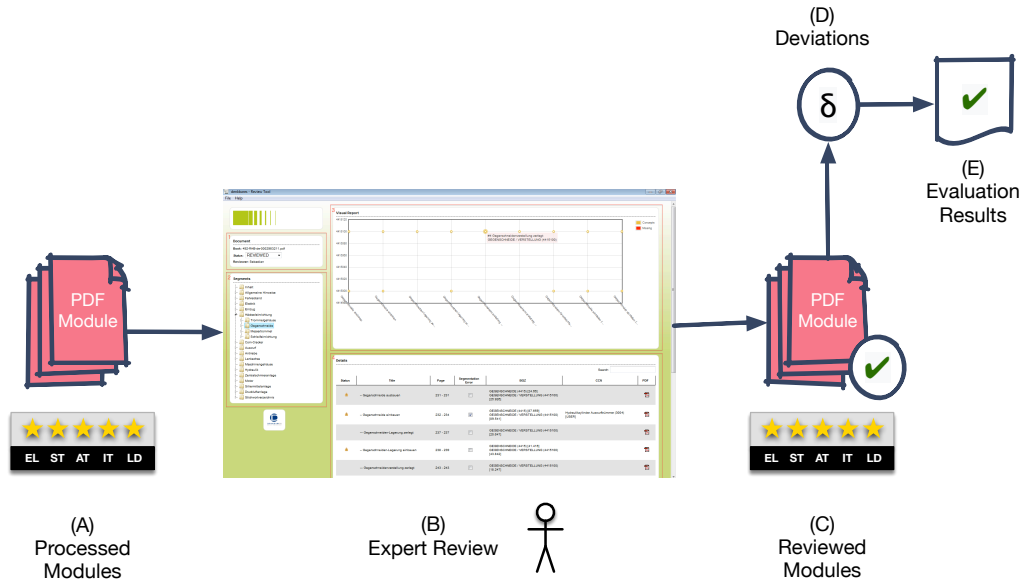


FIGURE 7.28 | Harvesting Technology: Evaluation Architecture.

covered different machines, document types, and languages. The randomly selected documents have then been reviewed by a domain expert (B). Finally, the formal evaluation was performed on basis of the deviations between the processed documents and the reviewed documents (D). The evaluation methods employed are described in detail in the next section.

### Evaluation Methods

The **5-STAR** semantic annotation aims on adding concept references to PDF-based modules. The automatically assigned annotations of the semantification process have been reviewed by a domain expert. The annotations of the reviewed modules served as gold standard, so the quality functions *precision*, *recall*, and *f-measure* can be determined in order to evaluate the results of the 5-STAR annotation step. The quality functions *precision*, *recall*, and *f-measure* are defined as follows

**Definition 7.7.1** (Annotation Precision). The annotation precision  $prec_{5-STAR}$  is defined as the number of correctly determined annotations with respect to all found annotations of a module  $p \in S_{pdf}$ :

$$prec_{5-STAR} = \frac{| \text{expected annotations} \cap \text{retrieved annotations} |}{| \text{retrieved annotations} |}.$$

An annotation is assumed as correct if it is also contained in the set of annotations of the corresponding reviewed module.

**Definition 7.7.2** (Annotation Recall). The annotation recall  $rec_{5-STAR}$  is defined as the number of correctly determined annotations with respect to all expected annotations of a module  $p \in S_{pdf}$ :

$$rec_{5-STAR} = \frac{| \text{expected annotations} \cap \text{retrieved annotations} |}{| \text{expected annotations} |}.$$

The set of expected annotations corresponds to the annotations of the reviewed module  $x \in S_{xml}$ .

**Definition 7.7.3** (Annotation F-Measure). The annotation f-measure is defined as the harmonic mean of  $prec_{5-STAR}$  and  $rec_{5-STAR}$ :

$$f = 2 * \frac{prec_{5-STAR} * rec_{5-STAR}}{prec_{5-STAR} + rec_{5-STAR}}.$$

### Evaluation Results

Table 7.4 shows the results, where the first three columns correspond to precision, recall, and f-measure and the fourth and fifth column show the number of corrections made by a domain expert. The minus sign (–) indicates the removal of an assigned concept and the plus sign (+) the addition of a missing concept.

Document	P	R	F	–	+
d1-SYS-de	0.67	1.00	0.80	5	0
d2-RHB-de	0.85	0.87	0.86	16	13
d3-RHB-fr	0.81	0.74	0.77	4	6
d4-RHB-de	1.00	0.92	0.96	0	3
d5-RHB-de	0.77	0.77	0.77	31	30
Overall	0.82	0.83	0.82	56	52

TABLE 7.4 | Harvesting Technology: Precision, Recall, F-Measure and Corrections.

Averaged over the evaluated documents the semantification architecture yields an f-measure of 82%. In these documents 108 corrections needed to be done by the domain expert. As the availability of a domain expert is critical, time needed for a correction was also estimated. The correction time was measured for randomly selected chapters from the documents above<sup>7</sup>. For each of the selected chapters the number of corrections have been measured as well as the total time needed for applying them (see Table 7.5) — an average correction time of 18 seconds per correction has been measured.

Document	# Corrections	∅ Time/Correction
d1-SYS-de (3)	2	22 s
d1-SYS-de (6)	1	20 s
d2-RHB-de (4)	5	8 s
d2-RHB-de (7)	10	16 s
d4-RHB-de (8)	1	28 s
d4-RHB-de (10)	1	16 s
d5-RHB-de (3)	10	20 s
d5-RHB-de (7)	12	14 s
Overall	42	18 s

TABLE 7.5 | Harvesting Technology: Measuring the correction effort.

#### 7.7.7 System Use

The semantified technical documentation is used in a productive after sales information system. The information system is based on a semantic search engine that works

<sup>7</sup>The French document was left out due to the absence of a French domain expert.

upon different taxonomies that describe products, components, and functions. For instance, a search for a specific machine and the component “Upper Discharge Spout” yields highly specific search results. The search results are grouped by books but refer to actual chapters like “3.11.4 Raising the upper discharge chute”. This example underlines the power of semantic search over semantified technical documents because the ambiguous terms “Upper Discharge Spout” and “Upper Discharge Chute” are automatically aligned. The basis for this alignment is the linking of the respective chapter with the “Upper Discharge Spout” concept from the domain ontology.

### **7.7.8 Summary**

This case study reported on an actual semantification project in the domain of harvesting technology. A semantification architecture for the processing of a large corpus of technical documents has been presented and evaluated. The evaluation results underline the practical applicability of the presented 5-STAR semantification approach. The technical documents that got semantified using the presented semantification architecture are available in a modern semantic information system that is in productive usage.



## 7.8 VW *ILTIS*

This case study reports on the semantification of scanned technical documents. The documents describe the operation and maintenance of the Volkswagen *ILTIS* vehicle.

### 7.8.1 Introduction

The semantification of a corpus of scanned documents is subject of this case study. The corpus contains technical documentation concerning the Volkswagen *ILTIS*, a vehicle that was used by the German Armed Forces back in the 1980s. The complete corpus of technical documents is available in forms of scanned images online. The documentation covers different skill levels of users and aspects of the vehicle. The documentation is entirely written in German language. This case study shows that the 5-STAR semantification approach is able to transform legacy documents to 5-STAR linkable representations that can be employed in a state-of-the-art semantic information system. Therefore, a 5-STAR semantification architecture has been applied.

### 7.8.2 Goals and Application Scenario

The purpose of this case study is the semantification of the scanned documentation which is not accessible, not modularized, and not semantically annotated. Hence, the goal of the case study is to produce a PDF-based data set that is compatible with a state-of-the-art semantic information system (see Section 7.8.6). The data set based on scanned resources, therefore gets semantically prepared. Hereby, the documents get subsequently semantified by running through respective semantification steps. This case study comprises an explicit nano structure recovery step that transforms the scanned images to text. For the fifth step in the semantification process a manually defined ontology is used.

### 7.8.3 Data Set Description

The corpus of technical documents for the Volkswagen *ILTIS* comprises twelve documents. The documentation has in total about 2,000 pages that are spread over different kinds of documents, e.g. repair manual, operation manual, spare parts, and trouble shooting descriptions. The documents address different target groups ranging from maintenance staff to end users what in turn influences the structure and the level of detail. The documents have been written in the 1980s using not computers but type writers. They have originally been published as written books and are today available as ordered collections of scanned images. As the documents have originally been written using type writers corresponding electronic source files are not available. Figure 7.29 shows scanned book pages of the VW *ILTIS* documentation.

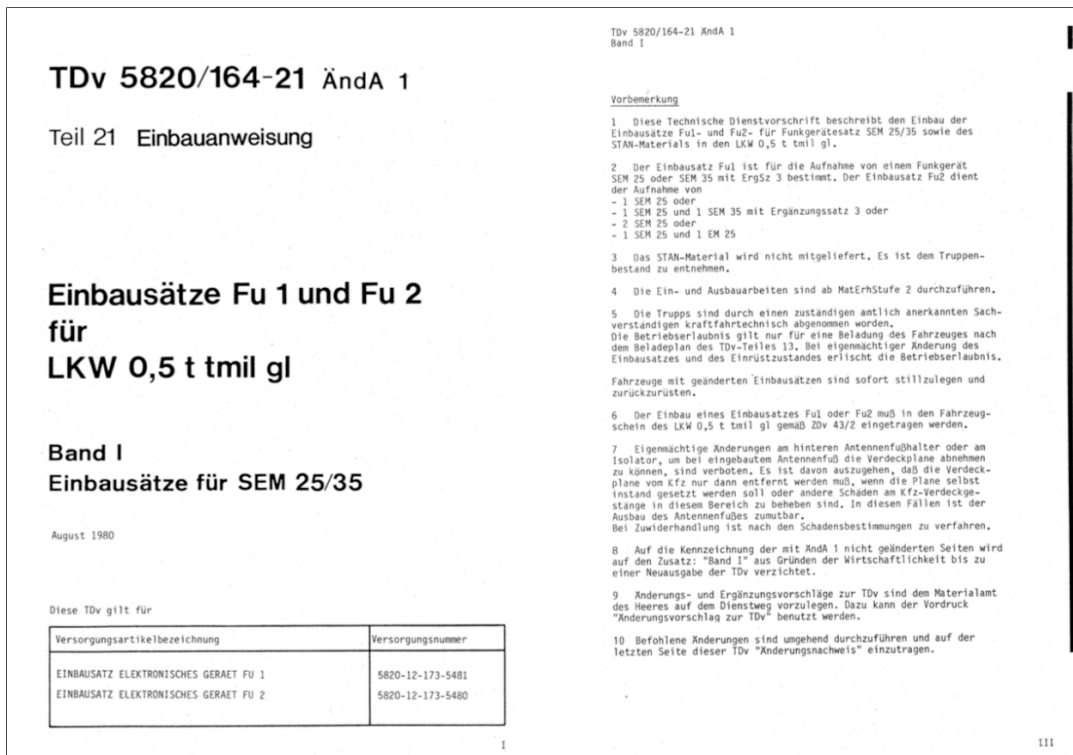


FIGURE 7.29 | VW ILTIS: Scanned Book Pages.

#### 7.8.4 The Semantification Process

The semantification architecture (see Figure 7.30) initially reads the scanned documents (0-STAR rating). A subsequent process applies Document Layout Analysis approaches to the images (A). Then, a subsequent processing step aims on recovering nano (B) and macro (C) structures. These recovered structures already allow to split the original document to smaller modules. Finally, a semantic annotation process (D) determines concepts from an external ontology for the semantic annotation of the modules. These modules finally yield 5-STARS in the maturity schema.

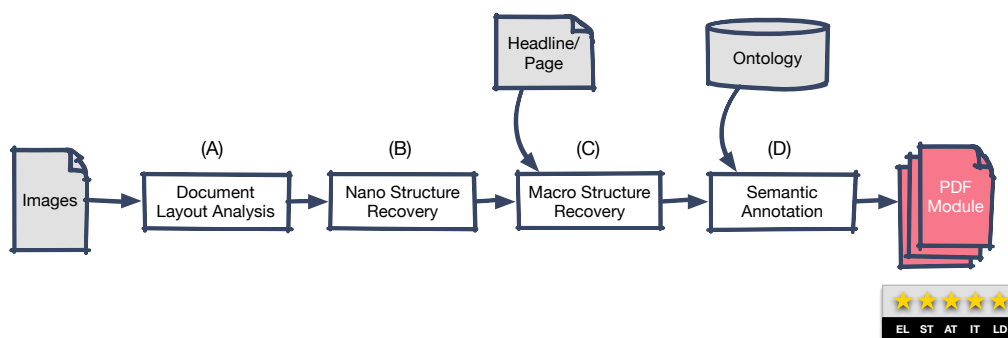


FIGURE 7.30 | VW ILTIS: Semantification Architecture.

For Document Layout Analysis (A), which especially comprises Optical Character Recognition, the open source command line application OCRmyPDF<sup>8</sup> has been applied. OCRmyPDF is based on Tesseract<sup>9</sup> and is able to add text layers to scanned

<sup>8</sup><https://github.com/jbarlow83/OCRmyPDF>

<sup>9</sup><https://github.com/tesseract-ocr/>

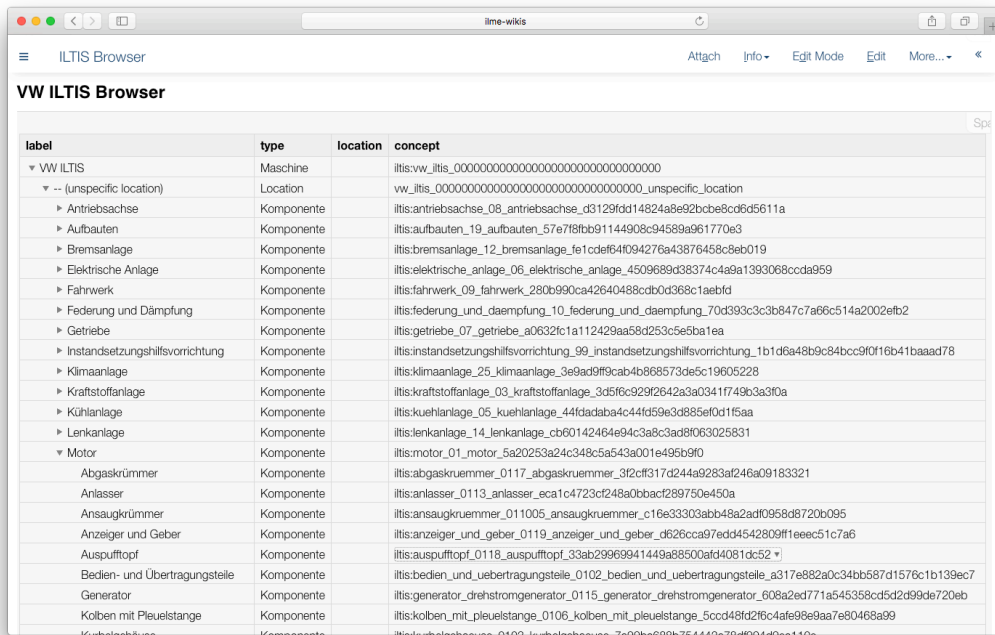
PDF documents. The nano structure recovery (B) is based on the open source tool pdf2xml [52]. An explicit micro structure recovery step has not been performed as the recovered text layers did not provide reasonable formatting information. Instead, the table of contents was manually extracted from the text layered and corresponding headlines matched on the target pages using fuzzy string matching techniques. The macro structure recovery (C) works upon these headline matches and employs the macro structure recovery mechanisms described in Section 4.5.3. The semantic annotation (D) is based on Probabilistic Explicit Semantic Analysis (PESA) described in Section 4.7.4 and uses a manually defined ontology (see Section 7.8.5).

### 7.8.5 Knowledge Resources

The ontology providing the concepts for the semantic annotation step has been manually extracted from a document that had already been processed by the Document Layout Analysis step. More specifically, the spare parts catalogue has been exploited to assemble an ontology that describes the technical composition of the Volkswagen ILTIS (see Figure 7.31).

GAPL	BEZEICHNUNG	BT-NR	BF-NR
01	MOTOR		Fiche 1
01	Motor, vollst.	01-001	1-B3
01	Motor, nackt	01-002	1-D3
0101	Motoraufhängung	01-002	1-D3
0102	Bedien- und Übertragungsteile	01-003	1-F3
0103	Kurbelgehäuse	01-004	1-H3
0104	Zylinderkopf	01-005	1-K3
0105	Kurbelwelle	01-006	1-N3
0106	Kolben mit Pleuelstangl.	01-006	1-N3
0107	Nockenwellenantrieb	01-007	1-D4
0108	Nockenwelle, Ventile, Ventilfedern	01-008	1-G4
010901	Ölpumpe	01-009	1-K4
010905	Ölfilter	01-010	1-M4
010911	Ölmeßstab, Zwischenwelle	01-010	1-M4
0110	Vergaser, vollst.	01-011	1-A5
0110	Vergaser-Einzelteile	01-012	1-C5
0110	Vergaser-Einzelteile	01-013	1-G5
011005	Ansaugkrümmer	01-014	1-K5
0112	Luftfiltersystem	01-015	1-M5
0112	Luftfiltersystem	01-016	1-A6
0113	Anlasser, vollst.	01-017	1-D6
0113	Anlasser-Einzelteile	01-018	1-F6
0115	Generator Antrieb+Befestigungsteile, Drehstromgenerator, vollst.	01-019	1-16
0115	Drehstromgenerator	01-020	1-K6
0117	Abgaskrümmer	01-021	1-N6
0118	Auspufftopf	01-021	1-N6
0119	Anzeiger und Geber	01-022	1-C7

FIGURE 7.31 | VW ILTIS: Spare Parts Catalogue providing Ontological Knowledge.



label	type	location	concept
▼ VW ILTIS	Maschine		iltis:vw_iltis_00000000000000000000000000000000
▼ -- (unspecific location)	Location		vw_iltis_00000000000000000000000000000000_unspecific_location
▶ Antriebsachse	Komponente		iltis:antriebsachse_08_antriebsachse_d3129fdd14824a8e92bcb8cd6d5611a
▶ Aufbauten	Komponente		iltis:aufbauten_19_aufbauten_57e7f8fbb91144908c94589a961770e3
▶ Bremsanlage	Komponente		iltis:bremsanlage_12_bremsanlage_fe1cdef64f094276a43876458c8eb019
▶ Elektrische Anlage	Komponente		iltis:elektrische_anlage_06_elektrische_anlage_4509689d38374c489a1393068ccda959
▶ Fahrwerk	Komponente		iltis:fahrwerk_09_fahrwerk_280b990ca42640488cb0d368c1aebfd
▶ Federung und Dämpfung	Komponente		iltis:federung_und_daempfung_10_federung_und_daempfung_70d393c3c3b847c7a66c514a2002efb2
▶ Getriebe	Komponente		iltis:getriebe_07_getriebe_a0632f1a112429aa58d253c5e5ba1ea
▶ Instandsetzungshilfsvorrichtung	Komponente		iltis:instandsetzungshilfsvorrichtung_99_instandsetzungshilfsvorrichtung_1b1d6a48b9c84bcc9f0f16b41baaad78
▶ Klimaanlage	Komponente		iltis:klimaanlage_25_klimaanlage_3e9ad9f9cab4b868573de5c19605228
▶ Kraftstoffanlage	Komponente		iltis:kraftstoffanlage_03_kraftstoffanlage_3d5f6c929f2e42a3a0341f749b3a3f0a
▶ Kühlanlage	Komponente		iltis:kuehlanlage_05_kuehlanlage_44fdadaba4c44fd59e3d885ef0d1f5aa
▶ Lenkanlage	Komponente		iltis:lenkanlage_14_lenkanlage_cb60142464e94c3a8c3ad8f063025831
▼ Motor	Komponente		iltis:motor_01_motor_5a20253a24c348c5a543a001e495b9f0
Abgaskrümmer	Komponente		iltis:abgaskruemmer_0117_abgaskruemmer_3f2cf317d244e9283af246a09183321
Anlasser	Komponente		iltis:anlasser_0113_anlasser_ea1c4723cf248a0bbact289750e450a
Ansaugkrümmer	Komponente		iltis:ansaugkruemmer_011005_ansaugkruemmer_c16e33303abb48a2adf0968d8720b095
Anzeiger und Geber	Komponente		iltis:anzeiger_und_geber_0119_anzeiger_und_geber_d626cca97dd4542809f11eeec51c7a6
Auspufftopf	Komponente		iltis:auspufftopf_0118_auspufftopf_33ab29969941449a88500afd4081dc52
Bedien- und Übertragungsteile	Komponente		iltis:bedien_und_uebertragungsteile_0102_bedien_und_uebertragungsteile_a317e882a0c34bb687d1576c1b139ec7
Generator	Komponente		iltis:generator_drehstromgenerator_0115_generator_drehstromgenerator_608a2ed771a545358cd5d2d99de720eb
Kolben mit Pleuelstange	Komponente		iltis:kolben_mit_pleuelstange_0106_kolben_mit_pleuelstange_5cc48fd2f6c4afe98e9aa7e80468a99

FIGURE 7.32 | VW ILTIS: Extracted Ontology.

The ontology contains information about the hierarchical structure of components of the VW ILTIS. Figure 7.32 is an excerpt of the extracted and compiled ontology that represents the component hierarchy.

### 7.8.6 System Use

The semantification process applied in this case study yields a semantified version of the VW ILTIS data set that is runnable in a state-of-the-art semantic information system. The presented semantification architecture has been implemented using CAPLAN. Figure 7.33 shows an excerpt of the defined semantification pipeline in CAPLAN. Figure 7.34 shows the semantified data set in the semantic information system “Service Mate”. More precisely the screenshot shows the search results for the semantic query “Fahrwerk” which is the German term for “chassis”. The top ranked search results represent modules describing different aspects of chassis and wheels, like technical details, maintenance work or the application of snow chains to the wheels. Service Mate opens a detail view of the search result when the user clicks on one of the search result cards.

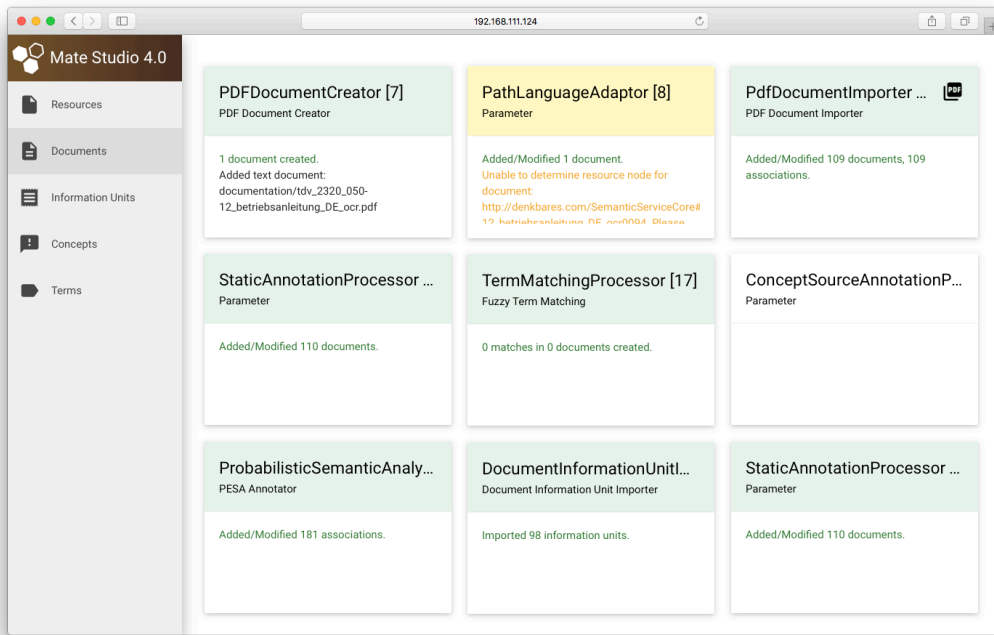


FIGURE 7.33 | VW ILTIS: Semantification Pipeline in CAPLAN.

Figure 7.35 shows Service Mate viewing a module. This demonstrates the fine grained recovery of modules even from scanned images. Additionally, the results of the entity recognition step are used to provide semantic links / search suggestions.

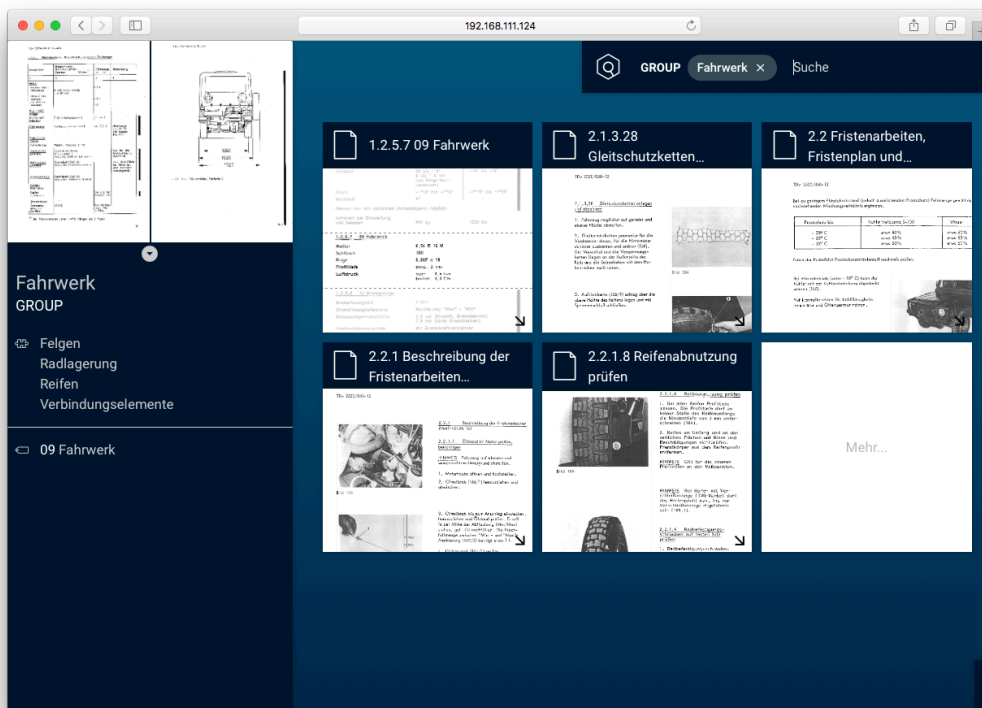


FIGURE 7.34 | VW ILTIS: Semantification Result (1/2).

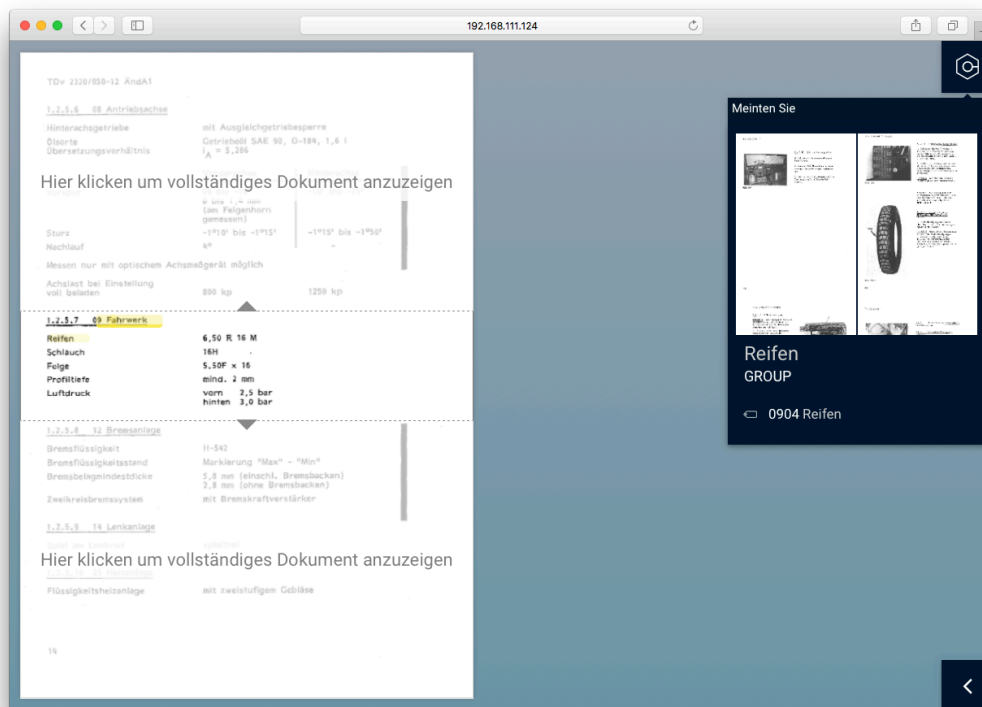


FIGURE 7.35 | VW ILTIS: Semantification Result (2/2).

### 7.8.7 Summary

This case study reported on the complete semantification of a collection of scanned technical documentation. The semantification comprised several semantification steps. In contrast to the other case studies the Document Layout Analysis task was significantly extended by Optical Character Recognition techniques that added text layers to the image based PDF documents. The recovered text was used as basis for subsequent semantification tasks, i.e. Macro Structure Recovery and Subject Analysis. A formal evaluation has not been performed on this data set. However, the results of this case study are runnable in the state-of-the-art semantic information system “Service Mate”. Considering the quality of the source data Macro Structure Recovery and Subject Analysis yield impressive results (see Section 7.8.6).

## Chapter 8

# Conclusion

This chapter concludes the presented work. First, a summary of the single chapters is given. Then, a discussion about the role of this work within the landscape of semantifying (technical) documents is given. The work closes with an outlook of interesting research questions considering the presented semantification approach.

## 8.1 Summary

### 8.1.1 Accessing Technical Documents

With the increasing complexity of machinery technical documentation has become an indispensable information source for service technicians. However, problem-oriented and targeted access to the different types of technical documents is not always possible in a straight-forward manner. Technical documents are special knowledge resources with manifold challenges arising from their specific characteristics. These characteristics can be exploited by state-of-the-art semantic information retrieval systems which are based on standardized technologies and make use of semantic information maintained in Linked (Enterprise) Data graphs. This work focuses on ontologies embedded in Linked (Enterprise) Data graphs that model machinery and documents. However, required semantic links between technical documents and concepts of Linked (Enterprise) Data graphs usually do not exist and the manual definition is a time-consuming and error-prone task. Hence, providing methods that support the process of linking reasonable parts of a technical document to ontological concepts describing their topic is the key research goal of this work.

### 8.1.2 Deep Semantics for Technical Documents

The latest developments in the fields of semantic information retrieval systems require technical documentation to be semantically prepared. The lack of a possibility to quickly assess the maturity of existing technical documents and the underlying information is a major problem in real-world projects. Therefore, this work introduced a novel 5-STAR maturity schema for technical documents. The maturity schema has five levels and a star is added on each level if documentation meets the respective requirements. This way structural accessibility on different granularity levels (1-STAR, 2-STAR, 3-STAR), modularization (3-STAR), identifiability and information typing (4-STAR) and linkability (5-STAR) can quickly be assessed.

Building upon the 5-STAR maturity schema TEKNO as a novel abstracting meta ontology for technical documents has been introduced. TEKNO provides a simple but powerful ontological vocabulary to semantically represent structural and rhetorical elements of technical documentation. Through its meta character it can easily be aligned to existing information models. Such alignments usually yield considerable improvements regarding accessibility. This is especially emphasized through a catalog of so called Core Documentation Entities that is part of the TEKNO ontology. Core Documentation Entities combine existing structural and rhetorical elements to deduct elements that carry strong technical knowledge. Examples for such Core Documentation Entities are repair instructions, component overviews, and measurement tables.

A plethora of information models supporting the authoring of proprietary and open technical documents exists, this work discussed the most considered open standards, namely: PI-Mod, iIRDS, DITA, DocBook and S1000D. For each information model the most important structural and rhetorical elements have been identified. Additionally, a mapping of the TEKNO ontology to the respective information model



has been described. All information models have been assessed according to the 5-STAR maturity schema.

### 8.1.3 Semantification of Technical Documents

Large corpora of technical documents that are not semantically prepared still exist in many companies. This limits their usage in state-of-the-art information systems. This is a severe problem as missing data quickly destroys the acceptance of novel information systems. Hence, there is a fundamental need for migrating legacy documents to formats that are compatible with modern information systems.

This work introduced a novel and holistic process that is able to transform documents in legacy formats like PDF to semantic representations. The industry-proven holistic process is based on five consecutive steps that build upon each other. The process is closely aligned with the 5-STAR maturity schema and assembled from established and novel methods. The novel methods are especially tailored to exploit the characteristics of technical documents.

The 1-STAR semantification aims on recovering basic document structures using methods from the fields of Document Layout Analysis. The 2-STAR semantification uses a novel micro structure classification approach to further detail document structure information. An interactive knowledge acquisition tool supports the development of respective classification knowledge bases. The 3-STAR semantification is concerned with recovering macro structure hierarchies of technical documents on basis of 2-STAR data. Additionally, similarity metrics are used to support deduplication and document alignment in multilingual corpora. The 4-STAR semantification uses Automatic Document Classification approaches which are employed to classify single macro structures (modules) with respect to information types. Finally, novel Subject Analysis and Subject Indexing approaches are used in order to realize the 5-STAR semantification. The presented approaches aim on adding concepts from an existing ontology as topical metadata to single macro structures (modules). This makes the respective modules usable in state-of-the-art semantic information systems. The approaches are knowledge-based and exploit domain knowledge concerning both, the characteristics of technical documents in focus and the underlying machines.

### 8.1.4 Implementation of a Semantification Architecture

The semantification architecture CAPLAN is a reference implementation of the 5-STAR semantification approach for technical documents. CAPLAN is a cloud-based semantification architecture that aims on enabling the accessible and scalable realization of semantification projects. Therefore, it provides a clean graphical user interface that is intended to be used by non-expert users. A plugin mechanism allows for the easy extension of CAPLAN with new semantification modules.

CAPLAN is accompanied by two supporting applications: TEKNO Studio and a review tool. Unlike CAPLAN that tackles the whole semantification process these applications are tools that are especially designed to support the knowledge acquisition for the 2-STAR semantification step and the review of 5-STAR semantification data. TEKNO Studio provides an interactive environment that displays PDF documents and allows to create and debug classification knowledge in forms of Set-Covering Models. The dedicated review tool allows to review and edit the results of the 5-STAR semantification process, i.e., results created by the CAPLAN application. The review tool is designed to support a targeted review by guiding a human reviewer to

potentially inconsistent data entries. Therefore, the semantic similarity of semantic annotations is visualized in a report, that allows to easily identify outlier concepts.

### 8.1.5 Practical Experiences

In four industrial projects the presented semantification approach or significant parts of it were used and successful experiences made:

- **Augmenting Spare Part Catalogues:**

A German software manufacturer for electronic spare part catalogues is currently transforming their product to an information portal for service technicians. Therefore, a respective semantification architecture integrates technical documents into the spare part application. As existing technical documents are usually not modularized and not semantically annotated the 5-STAR semantification approach is applied. The generic architecture is applied to data sets of different end-customers to link their data sets to spare part information. Unlike other semantification processes the fifth step in this architecture links the modules to concepts representing entries of the spare parts catalogue.

- **Earth Moving Technology:**

A German mechanical engineering company for earth moving technology offers its service technicians a novel information system for targeted access to required information. However, in order to provide access to relevant information, the information system requires technical documents to be semantically prepared. A semantification architecture was employed that transforms a large-scale PDF-based data set to a representation that is compatible with a semantic information system. The PDF resources, therefore, got semantically prepared by first splitting them into reasonable modules which then have been annotated with metadata that got extracted from discovered Core Documentation Entities.

- **Special Purpose Vehicles:**

A German mechanical engineering company for special purpose vehicles is providing its service technicians a novel semantic logistics and maintenance equipment. The respective software requires ontologies describing the vehicles. As such ontologies did not exist a semantification architecture was employed to populate multiple technical ontologies. Therefore, the semantification architecture exploited Core Documentation Entities in order to extract data that describe concepts and their relations.

- **Harvesting Technology:**

A German mechanical engineering company for harvesting technology has introduced a semantic information system. The underlying semantic information system offers service technicians targeted access to required information. However, in order to provide access to relevant information, the information system requires technical documents to be semantically prepared. Therefore, a 5-STAR semantification architecture was employed to transform of a large-scale PDF-based data set to a representation that is compatible with a semantic information system. The PDF resources, therefore, have been semantically prepared by first splitting them to reasonable modules which then have been annotated with concepts from an externally provided ontology.

In addition to these industrial projects three additional case studies underline the practical applicability of the approach. The benchmarking data sets of the S1000D

---

and the PI-Mod information models has been used to formally evaluate the semantification process. Additionally, a publicly available scanned technical documentation for the VW ILTIS has been semantified. The respective resources were authored in the 1980s using typewriters.

## 8.2 Discussion

This work presented a novel approach for the semantification of technical documents. The main contributions of this work are a semantic information model for technical documents and a holistic semantification process that has been implemented in the semantification architecture CAPLAN. These main contributions to semantification research are discussed in the following sections.

### 8.2.1 Self-Contained Semantification Approach

This work presented a self-contained semantification approach for technical documents, i.e., all aspects required for semantifying technical documents are considered. A semantification project typically requires a possibility to assess the semantic maturity of existing resources, a semantic target representation and a semantification architecture that is able to transform legacy data to the specified target representation. These aspects are covered in the presented approach as described in the following.

- **5-STAR Maturity Schema:**

The 5-STAR Maturity Schema described in this work is considered to be the fundamental basis for all semantification projects concerning technical documents. In practice, the early maturity assessment of existing resources is an important success factor in most projects as it is the basis for the definition of the project scope. As this requirement originates from sales the respective aspects are barely covered in scientific research. The presented 5-STAR maturity schema for technical documents, however, has been inspired by the more general idea of 5-STAR linked data. The assessment of data with respect to five consecutive maturity levels is one of the key aspects of Tim Berner-Lees vision of a linked data web [16]. In general, the linked data assessment is based on accessibility and linkability criteria. Although accessibility and linkability are also the main requirements for the assessment of technical documents, the general 5-STAR scheme does not consider all aspects of this special knowledge resource and consuming software applications. Hence, the requirements of the five maturity levels have been adapted to typical source formats of technical documents and (semantic) information retrieval use cases. The adapted maturity schema considers three different kinds of document structures: nano structures, micro structures, and macro structures, with nano structures being the most granular objects (e.g. tokens). Additionally, the annotation/linking with information types and additional concepts (topics) from an ontology is taken into account. These criteria allow to assess the semantic maturity of technical documents in an objective and standardized way. Consequently, an accurate assessment of technical documents can be given at an early phase in the project.

- **TEKNO Ontology:**

The next phase in a semantification project comprises the definition of the target representation. Nowadays, a lot of enterprise content management systems exist that have built in support for standardized information models for technical documents, like Apache DITA, PI-Mod, S1000D, iiRDS and DocBook. Additionally, a plethora of proprietary XML formats is available that are closely coupled to respective content management systems. The maturity assessments of the (de-facto) standard information models for technical documents have been discussed thoroughly in Chapter 3. In a nutshell, none of the existing standards received a real 5-STAR assessment, because of lacking granularity of

structures or limited linking possibilities. The degree of semantics, thus, must be considered to be limited for all standardized information models. Therefore, TEKNO as an abstracting semantic model for technical documents has been proposed in this work. TEKNO provides ontological elements for the description of structural and rhetorical elements of technical documents. Additionally, it allows to automatically derive Core Documentation Entities from existing structural and rhetorical document components. In order to make the TEKNO ontology broadly applicable and compatible to existing data sets, a mapping of the TEKNO classes and properties to the standardized information models Apache DITA, PI-Mod, S1000D, iiRDS, and DocBook has been described. The idea of such an abstracting ontology is not new. The main alternatives are the Document Components Ontology (DoCO) [46] and the SALT ontology [78]. Although these ontologies provide equal semantic expressiveness they have been designed for the representation of scientific articles. Mappings to existing information models for technicals do not exist. Thus, the usage of TEKNO as abstracting semantic representation layer is highly recommended for semantification projects within the scope of technical documents.

- **5-STAR Semantification:**

The final step in a semantification project is the actual preparation of the respective resources. Therefore, a semantification architecture must be defined. The semantification architecture describes which data (formats) must be imported, which of the 5-STAR semantification steps must be applied, and which export format(s) are desired. This work introduced a holistic approach that consists of five consecutive steps. The five semantification steps are closely aligned with the 5-STAR maturity model, i.e. processing data according to this process yields an additional star for each successfully performed step. Therefore, the underlying semantification process combines novel and established methods from Document Layout Analysis, Logical Document Structure Recovery, Macro Structure Recovery including Deduplication and Alignment of documents in multilingual corpora, Automatic Document Classification, and Subject Analysis. The respective approaches have been discussed thoroughly in Chapter 4. Summarized, this work has made major contributions in the fields of Logical Document Structure Recovery (2-STAR), Macro Structure Recovery (3-STAR) and Subject Analysis and Indexing (5-STAR). In these fields, novel, mostly knowledge-based, approaches have been proposed. All of these approaches exploit the special characteristics of technical documents and try to consider as much domain/background knowledge as possible. Unlike existing approaches found in literature, the presented methods build upon the fact that corporate style guides usually fix the appearance of technical documents for a specified period of time. Therefore, the definition of respective background knowledge yields major result improvements. Additionally, the presented approaches are designed to take uncertainty into account. This is an important aspect as the underlying data is often inconsistent and prone to errors. Typical examples comprise inaccurate formatting or ambiguously used terminology.

Scientific research concerning the problem of semantifying technical documents is rare. Bader and Oevermann [5] proposed a framework for the automatic classification and modularization of technical documents. The modularization seems to be rather loose and yields classified chunk blocks in the end. The detected chunk blocks are classified according to information types and technical components. In contrast to the

5-STAR semantification approach presented in this work the resulting modules (chunk blocks) must be considered to be rather small and potentially not self-contained. A fine grained hierarchy of nano, micro and macro structures is not available. In addition, the classification of small pieces of text from technical documents is usually much easier than determining the topic for a complete section as less terminology is involved which results in less potential classes/topics.

Gutierrez et al. [81] proposed a semantic framework for textual data enrichment. The approach tries to link textual resources to instances of WordNet [59]. Other systems claim to provide frameworks for the semantic integration of documents, but often consider only partial steps of the semantification problems.

### 8.2.2 On the Exploitation of Table of Contents

The first three steps of the described 5-STAR semantification approach aim on recovering macro structures (information units) from legacy technical documents. Instead of employing the 1-STAR, 2-STAR and 3-STAR semantification procedures, parsing the *Table of Contents* that is included in many technical documents, seems to be appealing as it contains many of the required information. However, detecting and parsing Table of Contents sections includes a lot of challenges, amongst others:

- **Detection:**

Although the identification of a Table of Contents section is an easy task for a human reader it is quite challenging for an automated process. Empirical studies that had preceded the development of the 5-STAR semantification approach tried to identify such sections using a combination of text patterns, page ranges (e.g. at the beginning of the document), frequency and distribution of special characters (e.g. "....."), and key words (e.g. "Index", "Table of Contents"). Although, this yielded good results for some documents, it did not work out for others. Thus, it was considered to be not generally applicable due to the heterogenous manner Table of Contents sections are defined in technical documents.

- **Other Indices:**

Additionally, in technical documents usually other indices exist, e.g. list of figures, and list of tables. Sometimes, technical documents also have redundant Table of Contents sections, e.g. a big one at the beginning of the book and smaller ones at the beginning of each chapter. The occurrence of multiple indices makes the automatic exploitation of included information more difficult.

- **Information Quality:**

The quality of information contained in Table of Contents sections is considered to be high. However, empirical studies showed that the quality of the contained information is not always consistent and useful. For instance, Table of Contents sections in technical documents might not reference page numbers but chapter numbers. This might be useful for service technicians but requires additional efforts for an automatic exploitation mechanism.

- **Missing Indices:**

Additionally, some types of technical documents do usually not even contain a Table of Contents section. This is especially true for spare parts catalogues. Hence, a macro structure recovery based on Table of Contents information is usually not applicable to a complete corpus of technical documents.

- **Definition of Background Knowledge:**

The presented 5-STAR semantification approach includes the definition of formalized background knowledge. This especially applies to the 2-STAR and 5-STAR semantification steps. The required knowledge is intended to be intuitive, even for non-technical users. Additionally, tools like TEKNO Studio support the knowledge acquisition in an interactive manner. The defined background knowledge is usually applicable for huge partitions of the corpus. In contrast, it is rather difficult to define background knowledge for the detection of table of contents sections that is applicable for multiple technical documents. Reasons are again the heterogeneity of technical documents.

The 5-STAR semantification approach presented in this work does not require the detection of Table of Contents sections. This makes the 5-STAR semantification approach applicable to the maturity of technical documents even when Table of Contents sections are missing or can hardly be detected automatically.

However, Table of Contents sections might be used in combination with the 5-STAR semantification approach. For instance, a macro structure recovered with the 5-STAR semantification approach might be checked against information extracted from a Table of Contents sections. This might help to identify errors or missing elements in the recovered macro structure. Additionally, information from a Table of Contents section might be used for the 2-STAR micro structure classification. The Set-Covering models used for the classification of untyped micro structures might be enhanced with background information about expected textual contents of headings.

### 8.2.3 Evaluation of the Practicability in Four Real-World Projects

The presented work has been used in the context of four real-life projects, i.e. the Spare Part Augmentation project, semantification projects for earth moving technology and harvesting technology as well as an ontology population project for special purpose vehicles. The respective projects were successfully finished and show promising results and satisfied customers. In all projects, the 5-STAR semantification approach or significant parts of it were easily adopted to the respective domain and corpora characteristics. The 5-STAR semantification approach and the accompanying applications CAPLAN, TEKNO Studio, and the review tool supported the efficient realization of the respective projects. This is mainly based on the easy to understand, commonly applicable, and tailorable character of the presented approach. The provided flexibility through the usage of knowledge-based methods allows to quickly adapt the process to new corpora and project requirements. This way, semantification projects can be achieved with reduced implementation effort. In essence, the presented approach transforms a classical task from the fields of Natural Language Processing to a Knowledge Engineering problem. This fact combined with the self-contained character of the approach allowed to quickly introduce colleagues at denkbare to semantification projects who did not have special knowledge in Natural Language Processing. In contrast to approaches that are based on Machine Learning methods the 5-STAR semantification process yields more predictable results which is an important aspect in real-world projects. Additionally, the general applicability of the 5-STAR semantification approach quickly yields first results that then can be iteratively and incrementally improved by further detailing the domain-specific background knowledge.

## 8.3 Outlook

This work provides a comprehensive introduction to the basics of a novel and holistic approach of semantifying technical documents. However, considering the plethora of technical documents that reside unsemantified, there are still interesting research questions to be explored. Additionally, new requirements and ideas arise from the broad practical application of the presented approach. This section outlines some of these ideas for improving and extending the 5-STAR semantification approach.

### 8.3.1 Fully Integrated Tool Chain

This work presented a tool chain that supports the 5-STAR semantification process. Namely, CAPLAN, a semantification architecture that allows for batch processing of technical documents, TEKNO Studio that supports the acquisition of classification knowledge for the 2-STAR semantification, and a dedicated review tool for 5-STAR data. The tools already allow to efficiently realize real-world semantification projects. However, the distribution of functionality over three different tools is not optimal as workflows get interrupted unnecessarily. For the future it is planned to fully integrate the complete tool chain in one application. Therefore, the functionality of TEKNO Studio and the review tool will be integrated in CAPLAN. So, the complete functionality required for semantification projects is available in a fully integrated cloud application.

### 8.3.2 Extending Core Documentation Entity Catalogue

The availability of Core Documentation Entities through the combination of structural and rhetorical document components is one of the most important aspects of this work. Core Documentation Entities dramatically increase the accessibility of technical documents. The current scope of Core Documentation Entities covers micro structures, which facilitates the targeted information extraction, e.g. for ontology population purposes (cf. see Case Study described in Section 7.6). This already is a considerable improvement within the scope of semantification projects. However, the Core Documentation Entity catalogue will be extended in the future in order to cover more application scenarios. The extension will mainly focus on increasing the granularity of Core Documentation Entities, i.e., the basic structural components will be nano structures. Having Core Documentation Entities available on the nano structure level applications like Question Answering [121, 145] directly from text will become possible.

### 8.3.3 Architecture Definition

The semantification pipelines presented in this work have been defined using CAPLAN. At the moment, CAPLAN distinguishes pipeline elements according to affected data node types, e.g. documents, concepts, or terms. The distributed definition of pipelines is not optimal in some application scenarios, especially when multiple data node types are involved. Therefore, it is planned to completely overhaul the pipeline definition functionality in CAPLAN. In the future, the pipeline definition might be available as a graphical flow chart editor. Then, it would be possible to define complete semantification architectures aided by a graphical editor. This graphical component would also indicate which results a single processing step produces and which compatible consuming processors are available. It is expected that this



will further increase the accessibility of semantification components for non-expert users.

### 8.3.4 Automatic Thesaurus Learning

This work presented a holistic semantification process that is able to modularize existing documentation and link the respective modules to Linked (Enterprise) Data graphs using concept annotations. The process is designed for batch processing and thus allows to semantify large corpora of technical documents without human interaction. Although all of the described case studies show promising results in real-world application scenarios it has been observed that the best results emerge from decent terminologies. However, in practice terminologies providing synonyms, hypernyms etc. often do not exist. General terminology lists like WordNet [59] or GermaNet [112] do usually not significantly affect the semantification results. Hence, future directions of this work comprise the integration of automatic thesaurus learning techniques into the semantification process. This way, inconsistently used terms can be aligned what in the end reduces the amount of potential annotation concepts and improves the linking quality. The evaluation of Automatic Thesaurus Learning methods will also comprise adaptations of existing methods in order to exploit available Core Documentation Entities.

### 8.3.5 Test, Analyze, Check

The semantification process presented in this work relies in large parts on knowledge-based methods. This is especially true for the 2-STAR semantification and the 5-STAR concept annotation. Additionally, CAPLAN as the reference implementation of the presented approach allows to configure different aspects of semantification architectures. Therefore, it is considered to be beneficial to provide automatic testing methods. Such methods should support both, unit tests for single aspects of an architecture and integration tests for the whole process pipeline. These tests should continuously and automatically check the results in order to prevent regression.

Additionally, TEKNO as abstracting ontology for information models provides new possibilities for checking already written documents for consistency. The available structural and rhetorical components as well as the catalogue of Core Documentation Entities allows to declaratively define consistency criteria, e.g. in forms of queries over the semantified data set. For the future, it is planned to provide a standard consistency catalogue for technical documents. This catalogue will then describe queries that for example check if every repair procedure is preceded by respective safety instructions.

### 8.3.6 Application to other Domains

Future directions of this work might also comprise the adaption to other domains. While the presented approach is in large parts designed for the semantification of technical documents the fundamental ideas might be applicable to other problem domains. The fundamental requirements for respective domains are the existence of basic guidelines respecting the appearance/formatting of documents and a certain degree of rhetorical structure regarding the textual contents. These requirements are possibly met in disciplines like medicine (e.g. medical reports) or law (e.g. statements of claim, contracts etc.). Additionally, the full potential of the presented work is exploited when existing documents get linked to concepts of a Linked Data graph.

For the medical domain the ICD classification is an important concept provider that has already been made available in the Linked Open Data cloud [192].

# Bibliography

- [1] Alexander Alexandrov et al. “The Stratosphere platform for big data analytics”. In: *The Very Large Data Bases Journal* 23.6 (2014), pp. 939–964.
- [2] David Andrzejewski and Xiaojin Zhu. “Latent Dirichlet Allocation with topic-in-set knowledge”. In: *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*. Association for Computational Linguistics. 2009, pp. 43–48.
- [3] Chidanand Apté, Fred Damerau, and Sholom M Weiss. “Automated learning of decision rules for text categorization”. In: *ACM Transactions on Information Systems (TOIS)* 12.3 (1994), pp. 233–251.
- [4] Sören Auer, Rene Pietzsch, and Jörg Unbehauen. “Datenintegration im Unternehmen mit Linked Enterprise Data”. In: *Linked Enterprise Data*. Springer, 2014, pp. 85–101.
- [5] Sebastian Bader and Jan Oevermann. “Semantic Annotation of Heterogeneous Data Sources: Towards an Integrated Information Framework for Service Technicians”. In: *Proceedings of the 13th International Conference on Semantic Systems*. ACM. 2017, pp. 73–80.
- [6] B. Baldwin. “CogNIAC: High Precision Coreference with Limited Knowledge and Linguistic Resources”. In: *Proceedings of the ACL Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*. 1997.
- [7] Michal Bali. *Drools JBoss Rules 5.0 Developer’s Guide*. Packt Publishing Ltd, 2009.
- [8] Alex Ball and Mansur Darlington. “Briefing Paper: The Adobe eXtensible Metadata Platform (XMP)”. In: *UKOLN research organization* (2007).
- [9] Joachim Baumeister and Jochen Reutelshoefer. “Developing knowledge systems with continuous integration”. In: *Proceedings of the 11th International Conference on Knowledge Management and Knowledge Technologies*. ACM. 2011, p. 33.
- [10] Joachim Baumeister, Jochen Reutelshoefer, and Frank Puppe. “KnowWE: A Semantic Wiki for Knowledge Engineering”. In: *Applied Intelligence* 35.3 (2011), pp. 323–344. URL: <http://dx.doi.org/10.1007/s10489-010-0224-5>.
- [11] Joachim Baumeister, Dietmar Seipel, and Frank Puppe. “Incremental Development of Diagnostic Set-Covering Models with Therapy Effects”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 11.Suppl. Issue 2 (2003), pp. 25–49. URL: <http://ki.informatik.uni-wuerzburg.de/papers/baumeister/2003-baumeister-SCM-ijufks.pdf>.
- [12] Joachim Baumeister et al. “KnowWE - A Wiki for Knowledge Base Development”. In: *The 8th Workshop on Knowledge Engineering and Software Engineering (KESE2012)*. [http://ceur-ws.org/Vol-949/kese8-05\\_04.pdf](http://ceur-ws.org/Vol-949/kese8-05_04.pdf), 2012.

- [13] Joachim Baumeister et al. “Linked Data City-Visualization of Linked Enterprise Data.” In: *Proceedings of the Conference "Lernen, Wissen, Daten, Analysen"*. 2016, pp. 145–152.
- [14] Sean Bechhofer et al. “OWL Web Ontology Language Reference”. In: *W3C Recommendation 10* (2004). URL: <http://www.w3.org/TR/owl-ref/>.
- [15] David Beckett, Tim Berners-Lee, and Eric Prud’hommeaux. *Turtle - Terse RDF Triple Language*. 2012. URL: <http://www.w3.org/TR/turtle/>.
- [16] Tim Berners-Lee. *Linked Data*. URL: <http://www.w3.org/DesignIssues/LinkedData.html>.
- [17] Tim Berners-Lee, Roy Fielding, and Larry Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986 (Standard). Internet Engineering Task Force, Jan. 2005. URL: <http://www.ietf.org/rfc/rfc3986.txt>.
- [18] Tim Berners-Lee, James Hendler, and Ora Lassila. “The Semantic Web”. In: *Scientific American* 284.5 (May 2001), pp. 34–43. ISSN: 0036-8733. URL: <http://www.sciam.com/2001/0501issue/0501berners-lee.html>.
- [19] Christian Bizer, Tom Heath, and Tim Berners-Lee. “Linked data – the story so far”. In: *International Journal on Semantic Web and Information Systems (IJSWIS)* 5.3 (2009), pp. 1–22.
- [20] David M Blei, Andrew Y Ng, and Michael I Jordan. “Latent Dirichlet Allocation”. In: *Journal of Machine Learning Research* 3 (2003), pp. 993–1022.
- [21] Andreas Blumauer. “Linked Data in Unternehmen. Methodische Grundlagen und Einsatzszenarien”. In: *Linked enterprise data*. Springer, 2014, pp. 3–20.
- [22] DCMI Usage Board. *DCMI metadata terms*. 2008.
- [23] Harold Boley et al. *RIF Core Dialect (Second Edition)*. Recommendation. W3C, 2013. URL: <http://www.w3.org/TR/2013/REC-rif-core-20130205/>.
- [24] Vinayak R. Borkar, Kaustubh Deshmukh, and Sunita Sarawagi. “Automatic Segmentation of Text into Structured Records.” In: *SIGMOD Conference*. Ed. by Sharad Mehrotra and Timos K. Sellis. ACM, 2001, pp. 175–186. ISBN: 1-58113-332-4. URL: <http://dblp.uni-trier.de/db/conf/sigmod/sigmod2001.html#BorkarDS01>.
- [25] Stefan Boschert and Roland Rosen. “Digital twinthe simulation aspect”. In: *Mechatronic Futures*. Springer, 2016, pp. 59–74.
- [26] Tim Bray et al. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. W3C Recommendation. 2008.
- [27] Thomas M Breuel. “High performance document layout analysis”. In: *Proceedings of the Symposium on Document Image Understanding Technology*. 2003, pp. 209–218.
- [28] Thomas M Breuel. “The hOCR microformat for OCR workflow and results”. In: *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*. Vol. 2. IEEE. 2007, pp. 1063–1067.
- [29] Thomas M Breuel. “The OCRopus open source OCR system”. In: *Document Recognition and Retrieval XV*. Vol. 6815. International Society for Optics and Photonics. 2008, 68150F.
- [30] Dan Brickley and Ramanathan V. Guha. “RDF Vocabulary Description Language 1.0: RDF Schema”. In: *W3C Recommendation 10* (2004). URL: <http://www.w3.org/TR/rdf-schema/>.

- [31] Dan Brickley, Ramanathan V Guha, and Brian McBride. *RDF Schema 1.1 – W3C Recommendation*. <http://www.w3.org/TR/rdf-schema>. 2014.
- [32] Eric Brill. “A simple rule-based part of speech tagger”. In: *Proceedings of the third conference on Applied natural language processing*. Association for Computational Linguistics. 1992, pp. 152–155.
- [33] Paul Browne. *JBoss Drools business rules*. Packt Publishing Ltd, 2009.
- [34] Paul Buitelaar and Philipp Cimiano. *Ontology learning and population: bridging the gap between text and knowledge*. Vol. 167. Ios Press, 2008.
- [35] Roldano Cattoni et al. “Geometric layout analysis techniques for document image understanding: a review”. In: *ITC-irst Technical Report 9703.09 (1998)*.
- [36] David Celjuska and Maria Vargas-Vera. “Ontosophie: A semi-automatic system for ontology population from text”. In: *Proceedings of the 3rd International Conference on Natural Language Processing (ICON)*. 2004.
- [37] François Chaumette. “Image moments: a general and useful set of features for visual servoing”. In: *IEEE Transactions on Robotics* 20.4 (2004), pp. 713–723.
- [38] Chaitanya Chemudugunta et al. *Modeling documents by combining semantic concepts with unsupervised statistical learning*. Springer, 2008.
- [39] Zhiyi Chi. “Statistical properties of probabilistic context-free grammars”. In: *Computational Linguistics* 25.1 (1999), pp. 131–160.
- [40] Freddy Y. Y. Choi. “Advances in domain independent linear text segmentation.” In: *ANLP*. 2000, pp. 26–33. URL: <http://dblp.uni-trier.de/db/conf/anlp/anlp2000.html#Choi00>.
- [41] Namyoun Choi, Il-Yeol Song, and Hyoil Han. “A survey on ontology mapping”. In: *ACM Sigmod Record* 35.3 (2006), pp. 34–41.
- [42] James Clark and Makoto Murata. *Relax NG specification*. 2001.
- [43] Sara Cohen et al. “XSEarch: A semantic search engine for XML”. In: *Proceedings of the 29th international conference on Very large data bases-Volume 29*. VLDB Endowment. 2003, pp. 45–56.
- [44] Alexis Conneau et al. *Very Deep Convolutional Networks for Natural Language Processing*. 2016. URL: <http://arxiv.org/abs/1606.01781>.
- [45] Alexandru Constantin, Steve Pettifer, and Andrei Voronkov. “PDFX: fully-automated PDF-to-XML conversion of scientific literature”. In: *Proceedings of the 2013 ACM symposium on Document engineering*. ACM. 2013, pp. 177–180.
- [46] Alexandru Constantin et al. “The Document Components Ontology (DoCO)”. In: *Semantic Web* 7.2 (2016), pp. 167–181.
- [47] Douglas Crockford. *The application/json media type for javascript object notation (json)*. 2006.
- [48] Morgan V Cundiff. “An introduction to the metadata encoding and transmission standard (METS)”. In: *Library Hi Tech* 22.1 (2004), pp. 52–64.
- [49] Fred J. Damerau. “A technique for computer detection and correction of spelling errors”. In: *Commun. ACM* 7.3 (Mar. 1964), pp. 171–176. ISSN: 0001-0782. DOI: 10.1145/363958.363994. URL: <http://doi.acm.org/10.1145/363958.363994>.

- [50] Margaret O Dayhoff. “A model of evolutionary change in proteins”. In: *Atlas of protein sequence and structure* 5 (1972), pp. 89–99.
- [51] Scott Deerwester et al. “Indexing by latent semantic analysis”. In: *Journal of the American society for information science* 41.6 (1990), pp. 391–407.
- [52] Hervé Déjean and Jean-Luc Meunier. “A system for converting PDF documents into structured XML format”. In: *International Workshop on Document Analysis Systems*. Springer, 2006, pp. 129–140.
- [53] Angelo Di Iorio et al. “Dealing with structural patterns of XML documents”. In: *Journal of the Association for Information Science and Technology* 65.9 (2014), pp. 1884–1900.
- [54] Petra Drewer and Wolfgang Ziegler. *Technische Dokumentation - Eine Einführung in die übersetzungsgerechte Texterstellung und in das Content Management*. Vogel Verlag, 2010.
- [55] Martin J. Dürst and Michel Suignard. *Internationalized Resource Identifiers (IRIs)*. 2007.
- [56] K Eberlein, R Anderson, and G Joseph. *Darwin Information Typing Architecture (DITA) 1.2 specification*. 2010. URL: <http://docs.oasis-open.org/dita/v1.2/spec/DITA1.2-spec.html>.
- [57] Jérôme Euzenat and Petko Valtchev. “Similarity-based ontology alignment in OWL-lite”. In: *Proceedings of the 16th European conference on artificial intelligence*. IOS press, 2004, pp. 323–327.
- [58] Ronen Feldman and James Sanger. *The Text Mining Handbook - Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2007, pp. I–XII, 1–410. ISBN: 978-0-521-83657-9.
- [59] Christiane Fellbaum, ed. *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press, 1998. ISBN: 978-0-262-06197-1.
- [60] Roy Fielding et al. *Hypertext transfer protocol–HTTP/1.1*. Tech. rep. 1999.
- [61] Katerina Frantzi, Sophia Ananiadou, and Hideki Mima. “Automatic recognition of multi-word terms: the C-value/NC-value method”. In: *International Journal on Digital Libraries* 3.2 (2000), pp. 115–130.
- [62] Sebastian Furth and Joachim Baumeister. “An ontology debugger for the semantic wiki KnowWE”. In: *10th Workshop on Knowledge Engineering and Software Engineering (KESE10)*. 2014.
- [63] Sebastian Furth and Joachim Baumeister. “Constructing Technical Knowledge Organizations from Document Structures”. In: *International Conference on Applications of Natural Language to Information Systems*. Springer, 2017, pp. 210–213.
- [64] Sebastian Furth and Joachim Baumeister. “On the Semantification of 5-STAR Technical Documentation”. In: *Proceedings of the LWA 2015 Workshops: KDML, FGWM, IR, and FGDB*. 2015, pp. 264–271.
- [65] Sebastian Furth and Joachim Baumeister. “Semantification of Large Corpora of Technical Documentation”. In: *Enterprise Big Data Engineering, Analytics, and Management*. IGI Global, 2016. URL: <http://www.igi-global.com/book/enterprise-big-data-engineering-analytics/145468>.
- [66] Sebastian Furth and Joachim Baumeister. “TELESUP-Textual Self-Learning Support Systems.” In: *FGWM’14: Proceedings of German Workshop of Knowledge and Experience Management at LWA’2014*. 2014, pp. 277–286.

- [67] Sebastian Furth and Joachim Baumeister. “Towards the Semantification of Technical Documents.” In: *FGIR'13: Proceedings of German Workshop of Information Retrieval (at LWA'2013)*. 2013, pp. 45–51.
- [68] Sebastian Furth, Volker Belli, and Joachim Baumeister. “The Review of Subject Analysis: A Knowledge-based Approach facilitating Semantic Search.” In: *Proceedings of the Conference "Lernen, Wissen, Daten, Analysen"*. 2016, pp. 165–174.
- [69] Sebastian Furth et al. “CAPLAN: An Accessible, Flexible and Scalable Semantification Architecture.” In: *Proceedings of the Conference "Lernen, Wissen, Daten, Analysen"*. 2016, pp. 177–185.
- [70] Sebastian Furth et al. “TEKNO: Preparing Legacy Technical Documents for Semantic Information Systems”. In: *Natural Language Processing and Information Systems: 22nd International Conference on Applications of Natural Language to Information Systems, NLDB 2017, Liège, Belgium, June 21-23, 2017, Proceedings*. Ed. by Flavius Frasincar et al. Cham: Springer International Publishing, 2017, pp. 429–434. ISBN: 978-3-319-59569-6. DOI: 10.1007/978-3-319-59569-6\_51. URL: [http://dx.doi.org/10.1007/978-3-319-59569-6\\_51](http://dx.doi.org/10.1007/978-3-319-59569-6_51).
- [71] Evgeniy Gabrilovich and Shaul Markovitch. “Computing semantic relatedness using Wikipedia-based explicit semantic analysis”. In: *Proceedings of the 20th international joint conference on artificial intelligence*. Vol. 6. 2007, p. 12.
- [72] Simon Garrod and Martin J Pickering. “Joint action, interactive alignment, and dialog”. In: *Topics in Cognitive Science* 1.2 (2009), pp. 292–304.
- [73] Alasdair Gilchrist. *Industry 4.0: the industrial internet of things*. Springer, 2016.
- [74] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. “Similarity search in high dimensions via hashing”. In: *Very Large Data Bases*. Vol. 99. 6. 1999, pp. 518–529.
- [75] Yoav Goldberg. “Neural network methods for natural language processing”. In: *Synthesis Lectures on Human Language Technologies* 10.1 (2017), pp. 1–309.
- [76] Charles F Goldfarb. *The SGML handbook*. Oxford University Press, 1990.
- [77] Martin Gollery. “Sequence and Genome Analysis”. In: *Clinical Chemistry* 51.11 (2005), pp. 2219–2219.
- [78] Tudor Groza et al. “SALT-Semantically Annotated LaTeX for Scientific Publications”. In: *The Semantic Web: Research and Applications*. Springer, 2007, pp. 518–532.
- [79] Frank E. Grubbs. “Procedures for Detecting Outlying Observations in Samples”. In: *Technometrics* 11.1 (1969), pp. 1–21.
- [80] Ramanathan Guha, Rob McCool, and Eric Miller. “Semantic search”. In: *Proceedings of the 12th international conference on World Wide Web*. ACM. 2003, pp. 700–709.
- [81] Yoan Gutiérrez, Sonia Vázquez, and Andrés Montoyo. “A semantic framework for textual data enrichment”. In: *Expert Systems with Applications* 57 (2016), pp. 248–269.

- [82] Eui-Hong Sam Han, George Karypis, and Vipin Kumar. “Text categorization using weight adjusted k-nearest neighbor classification”. In: *Pacific-asia conference on knowledge discovery and data mining*. Springer, 2001, pp. 53–65.
- [83] Steve Harris, Andy Seaborne, and E Prudhommeaux. *SPARQL 1.1 Query Language*. W3C Recommendation. W3C, 2013.
- [84] Marti A. Hearst. “TextTiling: segmenting text into multi-paragraph subtopic passages”. In: *Computational Linguistics* 23.1 (1997), pp. 33–64. ISSN: 0891-2017.
- [85] Gertjan van Heijst, Guus Schreiber, and Bob J. Wielinga. “Using explicit Ontologies in KBS Development”. In: *International Journal of Human-Computer Studies* 46.2-3 (1997), pp. 183–292. URL: <http://www.sciencedirect.com/science/article/B6WGR-45M91MP-V/2/229c0f079c683f3d7ae88cc279d0ec17>.
- [86] Gertjan van Heijst et al. “Foundations for a Methodology for Medical KBS Development”. In: *Knowledge Acquisition* 6.4 (1994), pp. 395–434. URL: <http://www.sciencedirect.com/science/article/B6WMS-45NJHVB-3/2/831e1c5aa6bd7026a6c0dfe3c6eb2acc>.
- [87] Steven Henikoff and Jorja G Henikoff. “Amino acid substitution matrices from protein blocks”. In: *Proceedings of the National Academy of Sciences* 89.22 (1992), pp. 10915–10919.
- [88] Pascal Hitzler, Rudolf Sebastian, and Markus Krötzsch. *Foundations of Semantic Web Technologies*. London: Chapman & Hall/CRC, 2009. ISBN: 9781420090505.
- [89] Pascal Hitzler et al., eds. *OWL 2 Web Ontology Language: Primer*. Available at <http://www.w3.org/TR/owl2-primer/>. W3C Recommendation, 27 October 2009.
- [90] J. Hobbs. “Resolving Pronoun References”. In: *Readings in Natural Language Processing*. California: Morgan Kaufmann, 1977.
- [91] Thomas Hofmann. “Probabilistic latent semantic indexing”. In: *ACM SIGIR Forum*. Vol. 51. 2. ACM, 2017, pp. 211–218.
- [92] Ian Horrocks et al. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. W3C Member Submission. World Wide Web Consortium, 2004. URL: <http://www.w3.org/Submission/SWRL>.
- [93] Bo Hu and Glenn Svensson. “A case study of linked enterprise data”. In: *International Semantic Web Conference*. Springer, 2010, pp. 129–144.
- [94] Anna Huang. “Similarity measures for text document clustering”. In: *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008)*, Christchurch, New Zealand, 2008, pp. 49–56.
- [95] Eero Hyvönen and Eetu Mäkelä. “Semantic autocompletion”. In: *The Semantic Web—ASWC 2006*. Springer, 2006, pp. 739–751.
- [96] Wouter IJntema et al. “A lexico-semantic pattern language for learning ontology instances from text”. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 15 (2012), pp. 37–50.
- [97] Antoine Isaac et al. “An empirical study of instance-based ontology matching”. In: *The Semantic Web*. Springer, 2007, pp. 253–266.



- [98] Robert Isele. “Methoden der Linked Data Integration”. In: *Linked Enterprise Data*. Springer, 2014, pp. 103–120.
- [99] Aminul Islam, Evangelos Milios, and Vlado Kešelj. “Text similarity using google tri-grams”. In: *Canadian Conference on Artificial Intelligence*. Springer, 2012, pp. 312–317.
- [100] Krzysztof Janowicz et al. “Five Stars of Linked Data Vocabulary Use”. In: *Semantic Web 5.3* (2014).
- [101] Jay J. Jiang and David W. Conrath. “Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy”. In: (1997).
- [102] Kyo Kageura and Bin Umino. “Methods of automatic term recognition: A review”. In: *Terminology 3.2* (1996), pp. 259–289.
- [103] Min-Yen Kan, Judith L. Klavans, and Kathleen McKeown. *Linear Segmentation and Segment Significance*. 1998. URL: <http://dblp.uni-trier.de/db/journals/corr/corr9809.html#cs-CL-9809020>.
- [104] Christopher Kennedy and Branimir Boguraev. “Anaphora for Everyone: pronominal anaphora resolution without a parser”. In: *Proceedings of the 16th International Conference on Computational Linguistics*. Copenhagen, Denmark, 1996.
- [105] Yoon Kim. *Convolutional Neural Networks for Sentence Classification*. 2014. URL: <http://arxiv.org/abs/1408.5882>.
- [106] Peter Kluegl et al. “UIMA Ruta: Rapid development of rule-based information extraction applications”. In: *Natural Language Engineering 22.01* (2016), pp. 1–40.
- [107] Graham Klyne and Jeremy J. Carroll. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. W3C Recommendation. Feb. 2004. URL: <http://www.w3.org/TR/rdf-concepts/>.
- [108] Youngjoong Ko. “A study of term weighting schemes using class information for text classification”. In: *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2012, pp. 1029–1030.
- [109] Stefan Kokkeliink and Roland Schwänzl. *Expressing qualified dublin core in RDF/XML*. 2001.
- [110] Markus Krötzsch et al. *OWL 2 Web Ontology Language Primer (Second Edition)*. Tech. rep. W3C, 2012. URL: <http://www.w3.org/TR/2012/REC-owl2-primer-20121211/>.
- [111] H. Kucera and W. N. Francis. *Computational analysis of present-day American English*. Providence, RI: Brown University Press, 1967.
- [112] Claudia Kunze and Lothar Lemnitzer. “GermaNet - representation, visualization, application.” In: *LREC*. European Language Resources Association, 2002. URL: <http://dblp.uni-trier.de/db/conf/lrec/lrec2002.html#KunzeL02>.
- [113] John Lafferty, Andrew McCallum, and Fernando Pereira. “Conditional random fields: Probabilistic models for segmenting and labeling sequence data”. In: *Proceedings of the 18. International Conference on Machine Learning, ICML*. Vol. 1. 2001, pp. 282–289.

- [114] Man Lan et al. “A comprehensive comparative study on term weighting schemes for text categorization with support vector machines”. In: *Special interest tracks and posters of the 14th international conference on World Wide Web*. ACM. 2005, pp. 1032–1033.
- [115] Guillaume Lazzara et al. “The SCRIBO module of the Olena platform: a free software framework for document image analysis”. In: *Document Analysis and Recognition (ICDAR), 2011 International Conference on*. IEEE. 2011, pp. 252–258.
- [116] Timothy Lebo et al. *PROV-O: The PROV Ontology: <http://www.w3.org/TR/prov-o>*. 2013.
- [117] Yuanguai Lei, Victoria Uren, and Enrico Motta. “Semsearch: A search engine for the semantic web”. In: *International Conference on Knowledge Engineering and Knowledge Management*. Springer. 2006, pp. 238–245.
- [118] V. Levenshtein. “Binary Codes Capable of Correcting Deletions, Insertions, and Reversals”. In: *Soviet Physics-Doklady* 10.8 (1966), pp. 707–710.
- [119] Håkon Wium Lie et al. *Cascading style sheets*. 2005.
- [120] Dekang Lin. “An information-theoretic definition of similarity.” In: *International Conference on Machine Learning, ICML*. Vol. 98. 1998, pp. 296–304.
- [121] Vanessa Lopez, Michele Pasin, and Enrico Motta. “Aqualog: An ontology-portable question answering system for the semantic web”. In: *European Semantic Web Conference*. Springer. 2005, pp. 546–562.
- [122] Minh-Thang Luong, Thuy Dung Nguyen, and Min-Yen Kan. “Logical structure recovery in scholarly articles with rich document features”. In: *Multimedia Storage and Retrieval Innovations for Digital Library Systems* 270 (2012).
- [123] Wei-Ying Ma and BS Manjunath. “A comparison of wavelet transform features for texture image annotation”. In: *Proceedings of International Conference on Image Processing 1995*. Vol. 2. IEEE. 1995, pp. 256–259.
- [124] Larry M Manevitz and Malik Yousef. “One-class SVMs for document classification”. In: *Journal of machine Learning research* 2.Dec (2001), pp. 139–154.
- [125] Christoph Mangold. “A survey and classification of semantic search approaches”. In: *International Journal of Metadata, Semantics and Ontologies* 2.1 (2007), pp. 23–34.
- [126] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press, 1999. ISBN: 0262133601.
- [127] Song Mao, Azriel Rosenfeld, and Tapas Kanungo. “Document structure analysis algorithms: a literature survey”. In: *Electronic Imaging 2003*. International Society for Optics and Photonics. 2003, pp. 197–207.
- [128] Lawrence McAfee. “Document classification using deep belief nets”. In: *CS224n, Sprint* 42 (2008).
- [129] Andrew McCallum, Dayne Freitag, and Fernando Pereira. “Maximum Entropy Markov Models for Information Extraction and Segmentation”. In: *Proceeding 17th International Conference on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 2000, pp. 591–598. URL: <http://citeseer.ist.psu.edu/mccallum00maximum.html>.

- [130] Andrew McCallum, Kamal Nigam, et al. “A comparison of event models for naive bayes text classification”. In: *AAAI-98 workshop on learning for text categorization*. Vol. 752. 1. Citeseer. 1998, pp. 41–48.
- [131] Johanna McEntyre and David Lipman. “PubMed: bridging the information gap”. In: *Canadian Medical Association Journal* 164.9 (2001), pp. 1317–1319.
- [132] Paul McNamee and James Mayfield. “Character n-gram tokenization for European language text retrieval”. In: *Information retrieval 7.1-2* (2004), pp. 73–97.
- [133] Babu M Mehtre et al. “Color matching for image retrieval”. In: *Pattern Recognition Letters* 16.3 (1995), pp. 325–331.
- [134] Xiangrui Meng et al. “Mllib: Machine learning in apache spark”. In: *JMLR* 17.34 (2016), pp. 1–7.
- [135] Dieter Merkl and Andreas Rauber. “Document classification with unsupervised artificial neural networks”. In: *Soft computing in information retrieval*. Springer, 2000, pp. 102–121.
- [136] J-L Meunier. “Optimized xy-cut for determining a page reading order”. In: *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*. IEEE. 2005, pp. 347–351.
- [137] Tomas Mikolov et al. “Distributed representations of words and phrases and their compositionality”. In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.
- [138] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *CoRR* (2013).
- [139] A. Miles and S. Bechhofer. *SKOS Simple Knowledge Organization System Reference. W3C Recommendation 18 August 2009*. 2009. URL: <http://www.w3.org/TR/2009/REC-skos-reference-20090818/>.
- [140] Robert Milne et al. “TIGER: Knowledge Based Gas Turbine Condition Monitoring”. In: *AI Communications* 9.3 (1996), pp. 92–108. ISSN: 0921-7126.
- [141] Ruslan Mitkov. “Robust Pronoun Resolution with Limited Knowledge.” In: *COLING-ACL*. Jan. 3, 2002, pp. 869–875. URL: <http://dblp.uni-trier.de/db/conf/acl/acl98.html#Mitkov98>.
- [142] Shunji Mori, Hirobumi Nishida, and Hiromitsu Yamada. *Optical character recognition*. John Wiley & Sons, Inc., 1999.
- [143] David Nadeau and Satoshi Sekine. “A survey of named entity recognition and classification”. In: *Lingvisticae Investigationes* 30.1 (2007), pp. 3–26. DOI: 10.1075/li.30.1.03nad. URL: <http://www.ingentaconnect.com/content/jbp/li/2007/00000030/00000001/art00002>.
- [144] George Nagy and Sharad Seth. *Hierarchical representation of optically scanned documents*. 1984.
- [145] Srinu Narayanan and Sanda Harabagiu. “Question answering based on semantic structures”. In: *Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics. 2004, p. 693.
- [146] Walsh Norman and Richard L Hamilton. *DocBook 5: The Definitive Guide*. O’Reilly Media, Inc., 2010.

- [147] Jan Oevermann. “Reconstructing Semantic Structures in Technical Documentation with Vector Space Classification.” In: *SEMANTiCS (Posters, Demos, SuCCESS)*. 2016.
- [148] Jan Oevermann and Wolfgang Ziegler. “Automated classification of content components in technical communication”. In: *Computational Intelligence* (2017).
- [149] Jan Oevermann and Wolfgang Ziegler. “Automated intrinsic text classification for component content management applications in technical communication”. In: *Proceedings of the 2016 ACM Symposium on Document Engineering*. ACM. 2016, pp. 95–98.
- [150] Lawrence O’Gorman. “The document spectrum for page layout analysis”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15.11 (1993), pp. 1162–1173.
- [151] OSGI Alliance. *OSGI Service Platform, Release 3*. IOS Press, Inc., 2003. ISBN: 1586033115.
- [152] T. Padma and P. Balasubramanie. “Knowledge based decision support system to assist work-related risk analysis in musculoskeletal disorder”. In: *Knowledge-Based Systems* 22.1 (2009), pp. 72–78. ISSN: 0950-7051. DOI: <http://dx.doi.org/10.1016/j.knosys.2008.07.001>.
- [153] Youngja Park, Roy J Byrd, and Branimir K Boguraev. “Automatic glossary extraction: beyond terminology identification”. In: *Proceedings of the 19th international conference on Computational linguistics-Volume 1*. Association for Computational Linguistics. 2002, pp. 1–7.
- [154] Dave Pawson. *XSL-FO: making XML look good in print*. O’Reilly Media, Inc., 2002.
- [155] M. Paziienza, M. Pennacchiotti, and F. Zanzotto. “Terminology Extraction: An Analysis of Linguistic and Statistical Approaches”. In: *Knowledge Mining Series: Studies in Fuzziness and Soft Computing*. Ed. by S. Sirmakessis. Springer Verlag, 2005.
- [156] Stefan Pletschacher and Apostolos Antonacopoulos. “The PAGE (page analysis and ground-truth elements) format framework”. In: *Pattern Recognition (ICPR), 2010 20th International Conference on*. IEEE. 2010, pp. 257–260.
- [157] M. Porter. “An Algorithm for Suffix Stripping”. In: *Program* 14.3 (1980), pp. 130–137. URL: [http://ontology.csse.uwa.edu.au/reference/browse\\_paper.php?pid=233281850](http://ontology.csse.uwa.edu.au/reference/browse_paper.php?pid=233281850).
- [158] David Martin Powers. *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*. 2011.
- [159] Frank Puppe. “Knowledge Formalization Patterns”. In: *PKAW 2000: Proceedings of the Pacific Rim Knowledge Acquisition Workshop*. Sydney, Australia, 2000.
- [160] Frank Puppe et al. “Clinical Experiences with a Knowledge-Based System in Sonography (SonoConsult)”. In: *Workshop on Current Aspects of Knowledge Management in Medicine (KMM05), Proceedings 3rd Conference Professional Knowledge Management - Experiences and Visions, Kaiserslautern, Germany*. 2005.

- [161] Daniel Ramage et al. “Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora”. In: *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*. Association for Computational Linguistics. 2009, pp. 248–256.
- [162] Cartic Ramakrishnan et al. “Layout-aware text extraction from full-text PDF of scientific articles”. In: *Source code for biology and medicine* 7 (2012).
- [163] Jean-Yves Ramel, Sébastien Busson, and Marie-Luce Demonet. “AGORA: the interactive document image analysis tool of the BVH project”. In: *Document Image Analysis for Libraries, 2006. DIAL’06. Second International Conference on*. IEEE. 2006, 11–pp.
- [164] Adwait Ratnaparkhi. “A maximum entropy model for part-of-speech tagging”. In: *Conference on Empirical Methods in Natural Language Processing*. 1996.
- [165] James Reggia. “Computer-Assisted Medical Decision Making”. In: *Applications of Computers in Medicine*. Ed. by Schwartz. IEEE, 1982, pp. 198–213.
- [166] Christian Reul, Uwe Springmann, and Frank Puppe. “LAREX: A Semi-automatic Open-source Tool for Layout Analysis and Region Extraction on Early Printed Books”. In: *Proceedings of the 2Nd International Conference on Digital Access to Textual Cultural Heritage*. DATeCH2017. Goettingen, Germany: ACM, 2017, pp. 137–142. ISBN: 978-1-4503-5265-9. DOI: 10.1145/3078081.3078097. URL: <http://doi.acm.org/10.1145/3078081.3078097>.
- [167] Jeffrey C. Reynar. “Statistical Models for Topic Segmentation.” In: *ACL*. Ed. by Robert Dale and Kenneth Ward Church. Association of Computer Linguistics, 1999. ISBN: 1-55860-609-2. URL: <http://dblp.uni-trier.de/db/conf/acl/acl1999.html#Reynar99>.
- [168] Roland Rosen et al. “About the importance of autonomy and digital twins for the future of manufacturing”. In: *IFAC-PapersOnLine* 48.3 (2015), pp. 567–572.
- [169] Yong Rui, Thomas S Huang, and Shih-Fu Chang. “Image retrieval: Current techniques, promising directions, and open issues”. In: *Journal of visual communication and image representation* 10.1 (1999), pp. 39–62.
- [170] Michael Rüßmann et al. “Industry 4.0: The future of productivity and growth in manufacturing industries”. In: *Boston Consulting Group* 9 (2015).
- [171] Melike Şah and Vincent Wade. “Automatic metadata extraction from multilingual enterprise content”. In: *Proceedings of the 19th ACM international conference on Information and knowledge management*. ACM. 2010, pp. 1665–1668.
- [172] G. Salton and C. Buckley. “Term-Weighting Approaches in Automatic Text Retrieval”. In: *Information Processing & Management* 24.5 (1988), pp. 513–523.
- [173] André Santos. “A survey on parallel corpora alignment”. In: *MI-STAR 2011* (2011), pp. 117–128.
- [174] Guus Schreiber et al. *Knowledge Engineering and Management - The CommonKADS Methodology*. 2nd ed. MIT Press, 2001.
- [175] Francesco Sclano and Paola Velardi. “Termextractor: a web application to learn the shared terminology of emergent web communities”. In: *Enterprise Interoperability II*. Springer, 2007, pp. 287–290.

- [176] Fabrizio Sebastiani. “Machine learning in automated text categorization”. In: *ACM computing surveys (CSUR)* 34.1 (2002), pp. 1–47.
- [177] Satoshi Sekine and Chikashi Nobata. “Definition, Dictionaries and Tagger for Extended Named Entity Hierarchy.” In: *LREC*. European Language Resources Association, 2004. URL: <http://dblp.uni-trier.de/db/conf/lrec/lrec2004.html#SekineN04>.
- [178] Fei Sha and Fernando Pereira. “Shallow parsing with conditional random fields”. In: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics. 2003, pp. 134–141.
- [179] Keith Shafer et al. “Introduction to persistent uniform resource locators”. In: *INET96* (1996).
- [180] Konstantin Shvachko et al. “The hadoop distributed file system”. In: *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*. IEEE. 2010, pp. 1–10.
- [181] Noam Slonim, Nir Friedman, and Naftali Tishby. “Unsupervised document classification using sequential information maximization”. In: *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2002, pp. 129–136.
- [182] Ray Smith. “An overview of the Tesseract OCR engine”. In: *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*. Vol. 2. IEEE. 2007, pp. 629–633.
- [183] T. Smith and M. Waterman. “Identification of Common Molecular Subsequences”. In: *Journal of Molecular Biology* 147.1 (1981), pp. 195–197.
- [184] Wee Meng Soon, Hwee Tou Ng, and Chung Yong Lim. “A Machine Learning Approach to Coreference Resolution of Noun Phrases.” In: *Computational Linguistics* 27.4 (2001), pp. 521–544. URL: <http://dblp.uni-trier.de/db/journals/coling/coling27.html#SoonNL01>.
- [185] Axel J Soto et al. “Similarity-based support for text reuse in technical writing”. In: *Proceedings of the 2015 ACM Symposium on Document Engineering*. ACM. 2015, pp. 97–106.
- [186] Manu Sporny et al. “JSON-LD 1.0: A JSON-based Serialization for Linked Data”. In: *W3C Recommendation* 16 (2014).
- [187] Achim Steinacker et al. *iiRDS Specification: intelligent information Request and Delivery Standard*. 2017. URL: [https://iirds.tekom.de/fileadmin/tekom\\_iirds/iiRDS\\_specification/index.html](https://iirds.tekom.de/fileadmin/tekom_iirds/iiRDS_specification/index.html).
- [188] Susie Stephens. “The enterprise semantic web”. In: *The Semantic Web*. Springer, 2007, pp. 17–37.
- [189] Fei Tao et al. “Digital twin-driven product design, manufacturing and service with big data”. In: *The International Journal of Advanced Manufacturing Technology* 94.9-12 (2018), pp. 3563–3576.
- [190] SL Ting, WH Ip, and Albert HC Tsang. “Is Naive Bayes a good classifier for document classification”. In: *International Journal of Software Engineering and Its Applications* 5.3 (2011), pp. 37–46.
- [191] Thanh Tran et al. “Ontology-based interpretation of keywords for semantic search”. In: *The Semantic Web*. Springer, 2007, pp. 523–536.

- [192] Tania Tudorache et al. “Using semantic web in ICD-11: three years down the road”. In: *International Semantic Web Conference*. Springer. 2013, pp. 195–211.
- [193] Jean Véronis. *Parallel Text Processing: Alignment and use of translation corpora*. Vol. 13. Springer Science & Business Media, 2013.
- [194] Norman Walsh and Leonard Muellner. *DocBook: the definitive guide*. Vol. 1. O’Reilly Media, Inc., 1999.
- [195] René Witte and Thomas Gitzinger. “Semantic Assistants – User-Centric Natural Language Processing Services for Desktop Clients”. In: *3rd Asian Semantic Web Conference (ASWC 2008)*. Vol. 5367. LNCS. Bangkok, Thailand: Springer, 2009, pp. 360–374. URL: <http://rene-witte.net/semantic-assistants-aswc08>.
- [196] Dengsheng Zhang and Guojun Lu. “Shape-based image retrieval using generic Fourier descriptor”. In: *Signal Processing: Image Communication* 17.10 (2002), pp. 825–848.
- [197] Tong Zhang and Frank J Oles. “Text categorization based on regularized linear classification methods”. In: *Information retrieval* 4.1 (2001), pp. 5–31.
- [198] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. *Character-level Convolutional Networks for Text Classification*. 2015. URL: <http://arxiv.org/abs/1509.01626>.
- [199] Ye Zhang and Byron C. Wallace. *A Sensitivity Analysis of (and Practitioners’ Guide to) Convolutional Neural Networks for Sentence Classification*. 2015. URL: <http://arxiv.org/abs/1510.03820>.
- [200] Ziqi Zhang et al. *A comparative evaluation of term recognition algorithms*. 2008.
- [201] Wolfgang Ziegler. “Content Management und Content Delivery. Powered by PI-Class”. In: *Tagungsband zur tekom Jahrestagung*. 2015.
- [202] Wolfgang Ziegler and Stephan Steurer. *Mit PI-Mod dokumentieren. Standardisiertes Informationsmodell für den Maschinen- und Anlagenbau*. 2010.